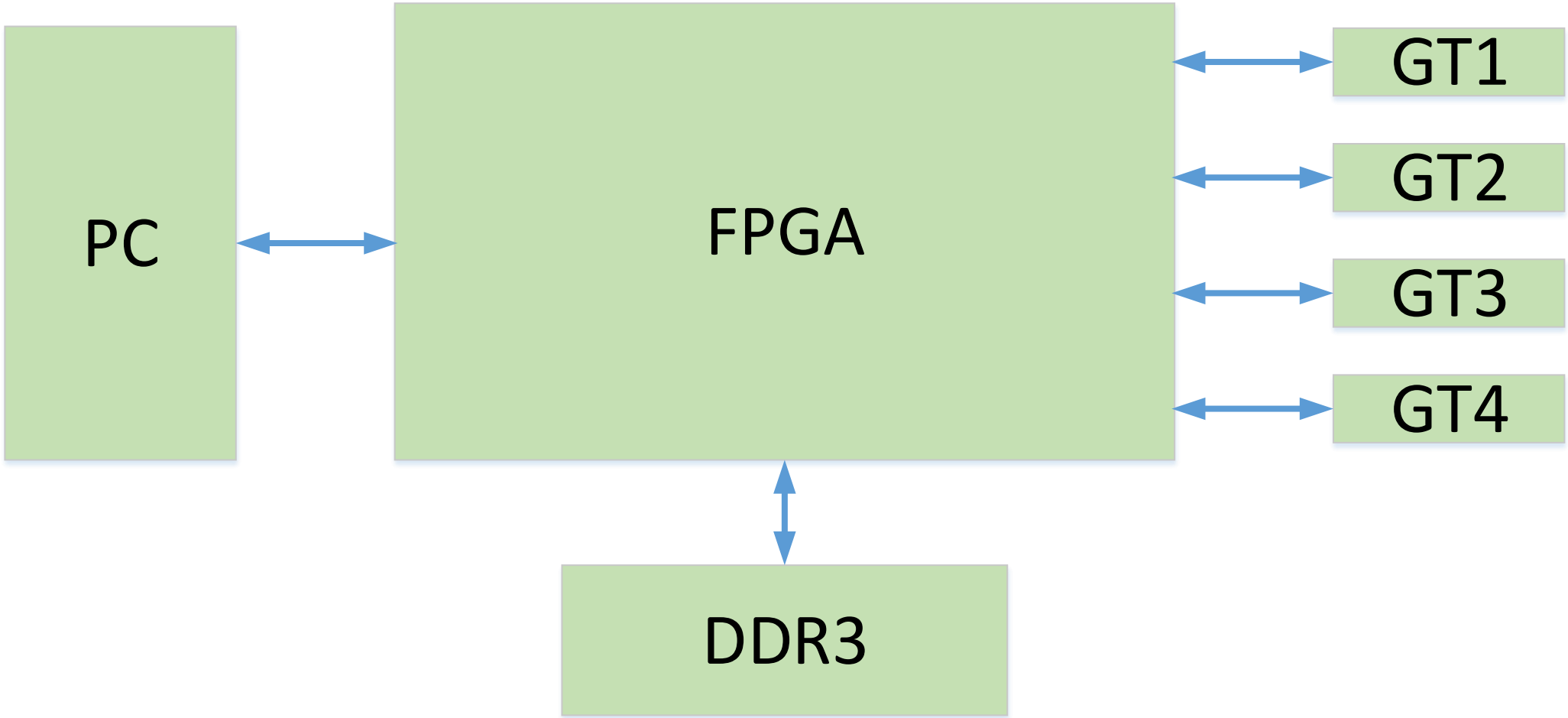
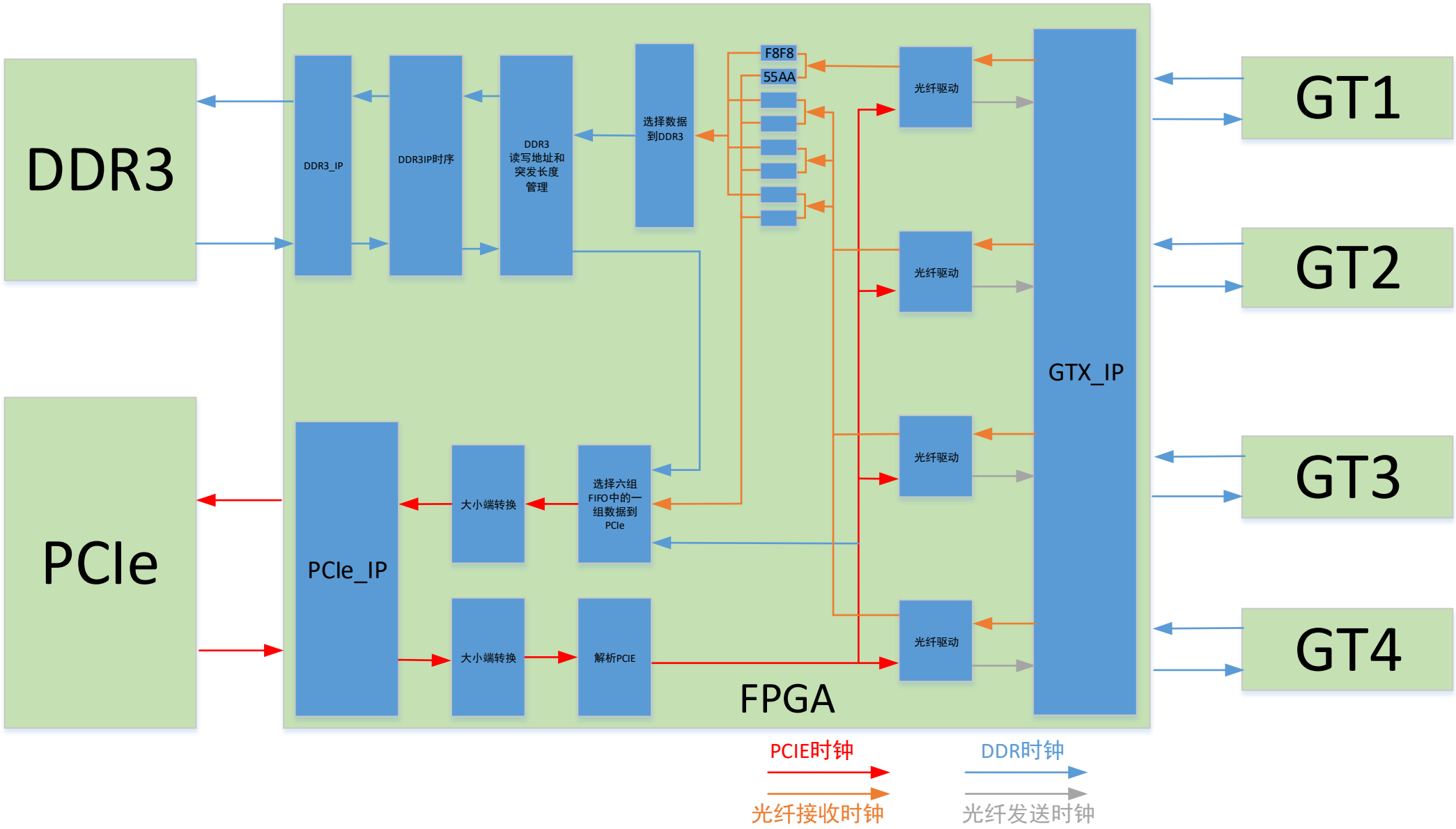
光纤项目

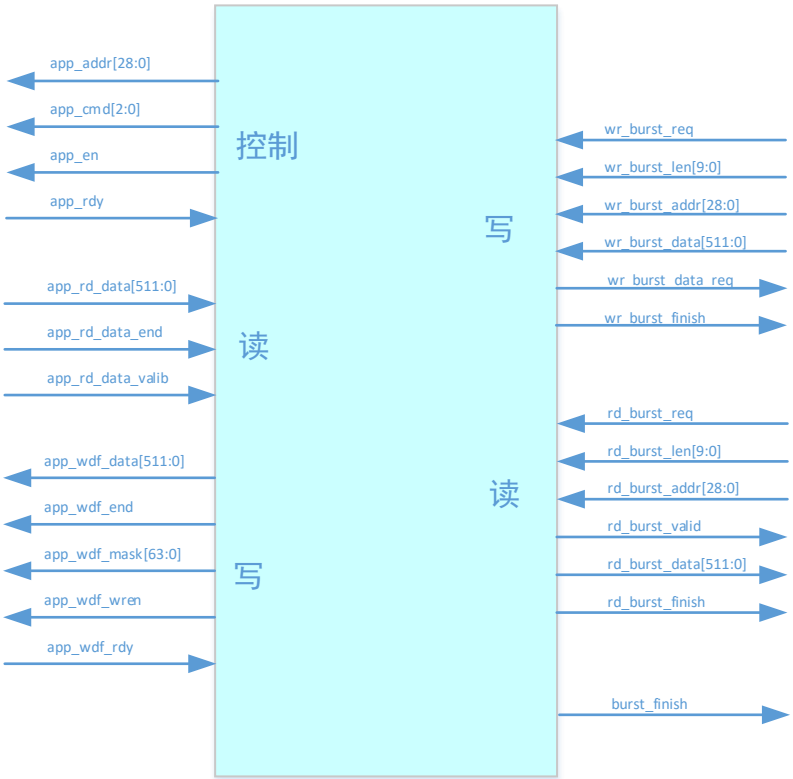
上位机通过PCIe把数据发送给FPGA，FPGA打包后通过光纤模块发送出去，同时FPGA保存光纤过来的数据到DDR3中，当DDR3中的数据存够一定数量，把DDR3中的数据通过PCIe发送给上位机



光纤项目功能框图


DDR3
PCIe
DDR3_IP
DDR3IP时序
DDR3
读写地址和
突发长度
管理
选择数据
到DDR3
F8F8
55AA
光纤驱动
光纤驱动
光纤驱动
光纤驱动
GTX_IP
GT1
GT2
GT3
GT4
PCIe_IP
大小端转换
选择六组
FIFO中的一
组数据到
PCIe
大小端转换
解析PCIE
FPGA
PCIE时钟
光纤接收时钟
DDR时钟
光纤发送时钟

模块名：mem_burst_xilinx
例化名：u_mem_burst
作用　：按照DDR3_IP的驱动时序
　　把数据发送给DDR3的IP



框图
下一页

备注1：大小端转换

```
genvar  ww;
generate
   for (ww=0 ; ww<16; ww=ww+1 ) begin
      assign   pcie_mty_rx[ww*8 +7 - :8] = pcie_mty_rx_fifo[127-ww*8 - : 8];
   end
Endgenerate
```

上述公式等价于：

```
assign   pcie_mty_rx[0*8 +7 - :8] = pcie_mty_rx_fifo[127-0*8 - : 8];
assign   pcie_mty_rx[1*8 +7 - :8] = pcie_mty_rx_fifo[127-1*8 - : 8];
assign   pcie_mty_rx[2*8 +7 - :8] = pcie_mty_rx_fifo[127-2*8 - : 8];
assign   pcie_mty_rx[3*8 +7 - :8] = pcie_mty_rx_fifo[127-3*8 - : 8];
                                    .
                                    .
                                    .
assign   pcie_mty_rx[14*8 +7 - :8] = pcie_mty_rx_fifo[127-14*8 - : 8];
assign   pcie_mty_rx[15*8 +7 - :8] = pcie_mty_rx_fifo[127-15*8 - : 8];

// 解释   ：  [7 - : 8]
// 等价于：  [a - : b ]  == [a : a - b ] == [7 :0 ]    // 因为是从0 开始数的 ，7 6 5 4 3 2 1 0

上述公式等价于：
assign   pcie_mty_rx[ 7  :  0] = pcie_mty_rx_fifo [127 : 120];
assign   pcie_mty_rx[15 :  8] = pcie_mty_rx_fifo [119 : 112];
assign   pcie_mty_rx[23 :16] = pcie_mty_rx_fifo [111 : 104];
assign   pcie_mty_rx[31 :24] = pcie_mty_rx_fifo [103 :  96];
                                    .
                                    .
                                    .
```

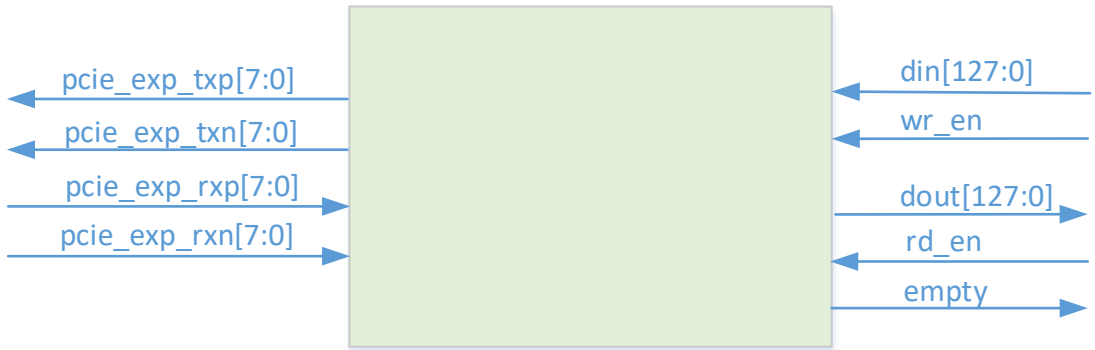
备注2：hot2bin

```
//**********************
//   判断输入数据中，第几位被值1 (低位优先级高 )
//**********************/
//举例：输入数据例子：
                    if(din==5'b00001)
                    if(din==5'b00010)
                    if(din==5'b00100)
                    if(din==5'b01000)
                    if(din==5'b10000)
//计算：先计算哪一位有效：
                    sel[0 ]=din[0    ];
                    sel[1 ]=sel[0 :0]==0 && din[1 ];
                    sel[2 ]=sel[1 :0]==0 && din[2 ];
                    sel[3 ]=sel[2 :0]==0 && din[3 ];
                    sel[4 ]=sel[3 :0]==0 && din[4 ];
                    sel[5 ]=sel[4 :0]==0 && din[5 ];
//计算并输出：判断并输出
                    dout_bin[0] = sel[1 ] | sel[3 ] | sel[5 ] ;
                    dout_bin[1] = sel[2 ] | sel[3 ] ;
                    dout_bin[2] = sel[4 ] | sel[5 ] ;
 //参考：十进制表达法和二进制表达法
     5'd1  ==  3'b001
     5'd2  ==  3'b010
     5'd3  ==  3'b011
     5'd4  ==  3'b100
     5'd5  ==  3'b101
```

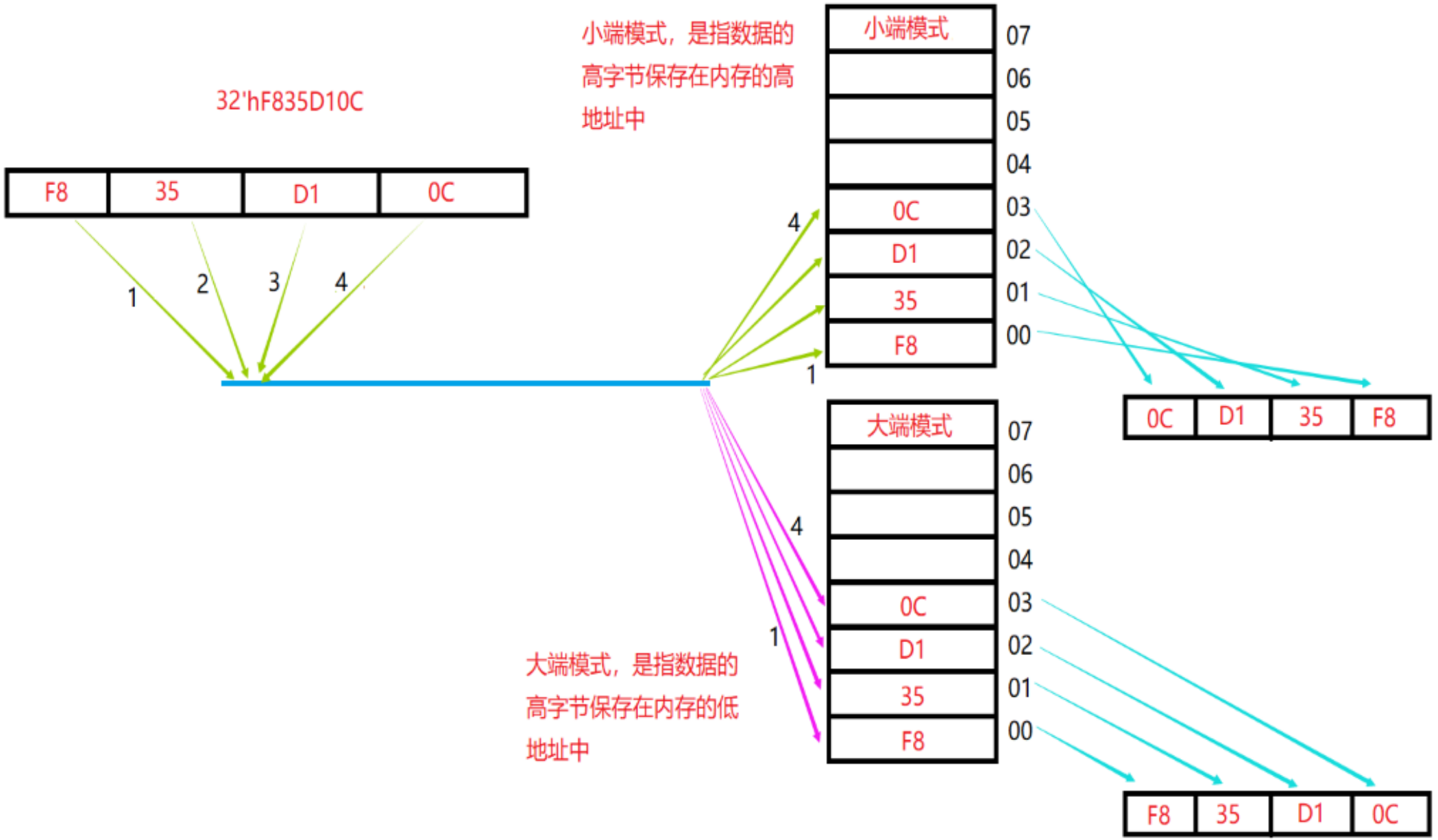
备注3：判断读取哪个ID

```
genvar   ii;
generate
   for (ii=0 ;  ii<5;  ii=ii+1  ) begin
      assign   flag_id[ii] = flag_ctrl && din[110:96] ==i i;
   end
Endgenerate

flag_id[0] = din[110:96] ==0;
flag_id[1] = din[110:96] ==1;
flag_id[2] = din[110:96] ==2;
flag_id[3] = din[110:96] ==3;
flag_id[4] = din[110:96] ==4;
```

PCIe_IP核


pcie_exp_txp[7:0]
pcie_exp_txn[7:0]
pcie_exp_rxp[7:0]
pcie_exp_rxn[7:0]
din[127:0]
wr_en
dout[127:0]
rd_en
empty

框图

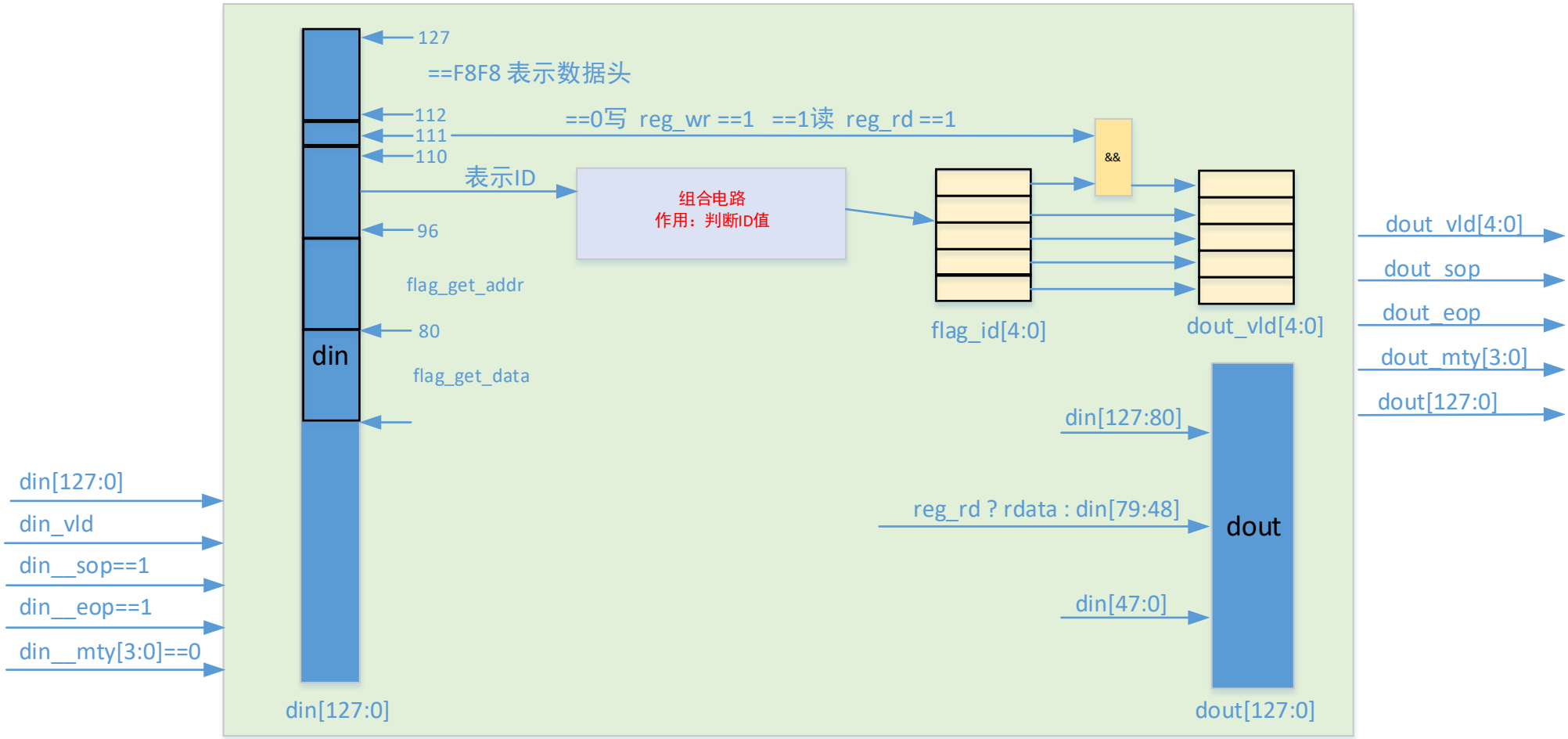


所以：32'hf835d10c这个数据从FPGA中发送到大端CPU是不需要转换的，发小端CPU则需要转换

框图

下一页

模块名　　：cfg_pack_analy_rx
模块例化名字：u_cfg_rxb
作用　　：解析PCI包头16'hF8F8



框图

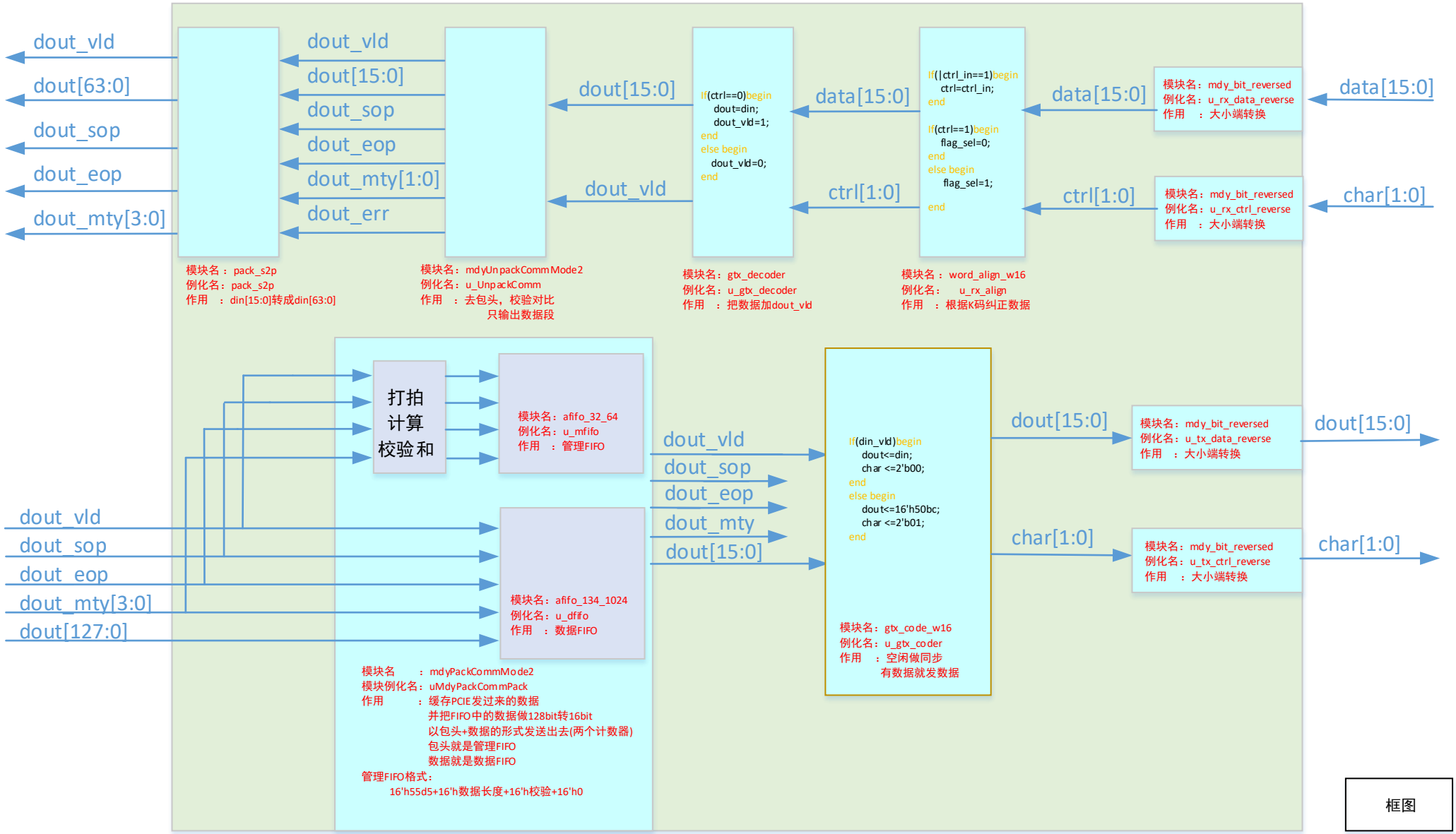
判断读取哪个ID

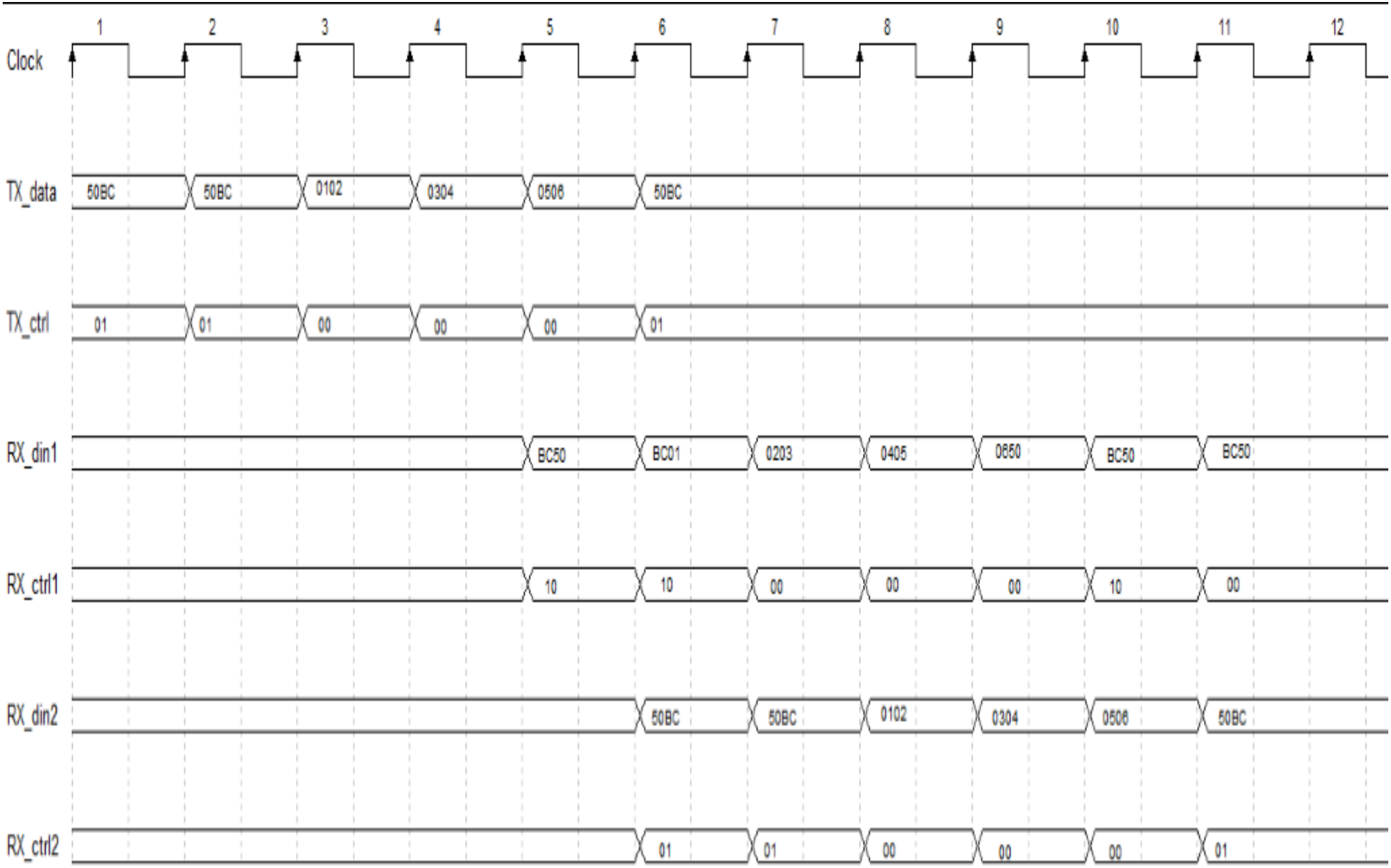
```
genvar   ii;
generate
   for (ii=0 ;  ii<5;  ii=ii+1  ) begin
       assign   flag_id[ii] = flag_ctrl && din[110:96] ==i i;
   end
Endgenerate
```

等价于：

```
flag_id[0] = din[110:96] ==0;
flag_id[1] = din[110:96] ==1;
flag_id[2] = din[110:96] ==2;
flag_id[3] = din[110:96] ==3;
flag_id[4] = din[110:96] ==4;
```

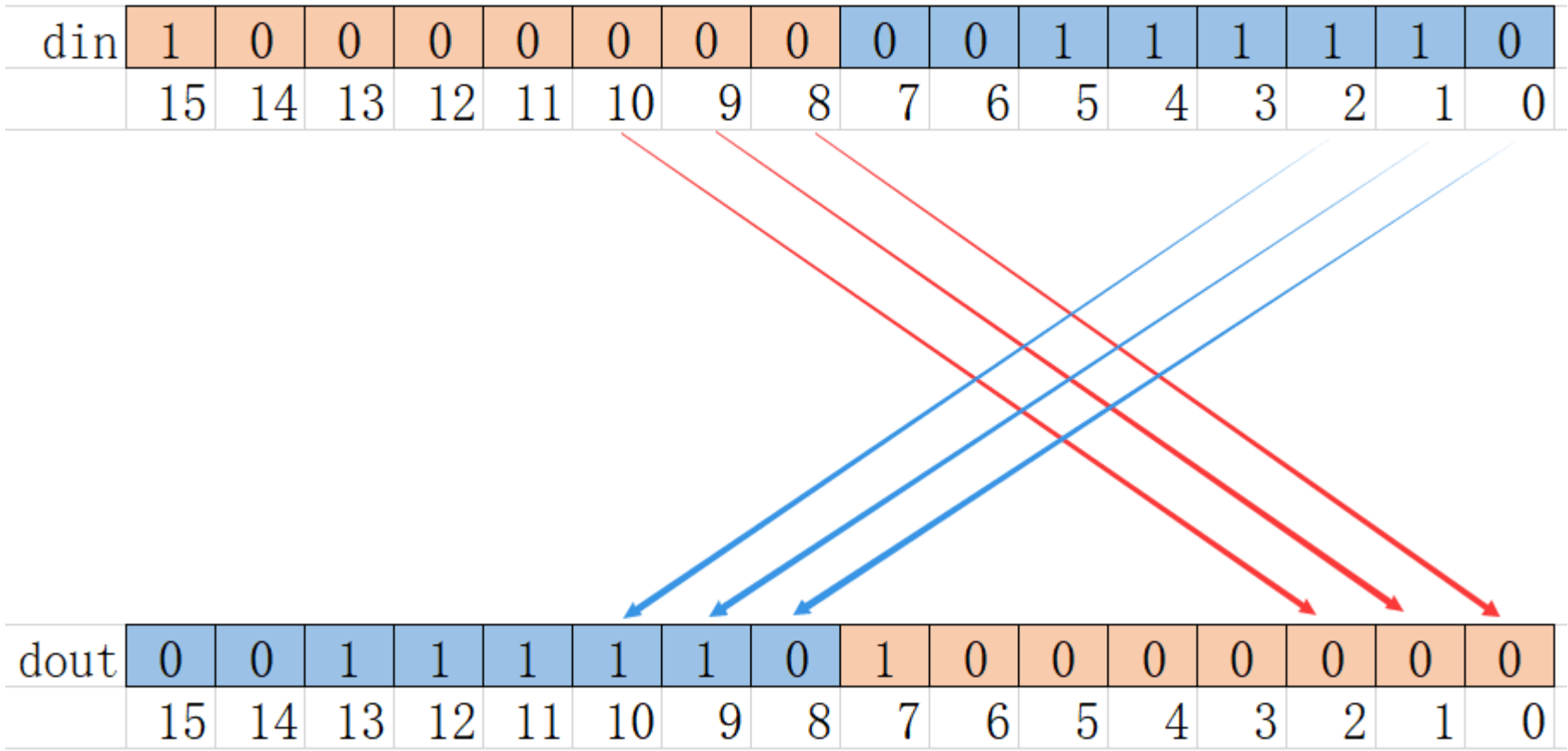
模块名：gtxPackAndCodeMode2
例化名：u_code
作用　：把PCIE发送过来的数据，打包发送给光纤IP。
　　把光纤IP发过来包文，拆包并发送出去




1
2
3
4
5
6
7
8
9
10
11
12
Clock
TX_data
50BC
50BC
0102
0304
0506
50BC
TX_ctrl
01
01
00
00
00
01
RX_din1
BC50
BC01
0203
0405
0650
BC50
BC50
RX_ctrl1
10
10
00
00
00
10
00
RX_din2
50BC
50BC
0102
0304
0506
50BC
RX_ctrl2
01
01
00
00
00
01

上一页

```
for( ii =0; ii<2; ii++ )
  for( jj=0; jj<8; jj++)
     dout_tmp [ ii*8 + jj ] = din[ (2-ii-1)*8+ jj ];

 ii=0;                                ii=1;
  dout_tmp [0] = din[ 8+0 ];          dout_tmp [8] = din[ 0 ];
  dout_tmp [1] = din[ 8+1 ];          dout_tmp [9] = din[ 1 ];
  dout_tmp [2] = din[ 8+2 ];          dout_tmp [10] = din[ 2 ];
  dout_tmp [3] = din[ 8+3 ];          dout_tmp [11] = din[ 3 ];
  dout_tmp [4] = din[ 8+4 ];          dout_tmp [12] = din[ 4 ];
  dout_tmp [5] = din[ 8+5 ];          dout_tmp [13] = din[ 5 ];
  dout_tmp [6] = din[ 8+6 ];          dout_tmp [14] = din[ 6 ];
  dout_tmp [7] = din[ 8+7 ];          dout_tmp [15] = din[ 7 ];
```



上一页

大小端转换

```
genvar  ww;
generate
   for (ww=0 ; ww<16; ww=ww+1  ) begin
       assign   pcie_mty_rx[ww*8 +7 - :8] = pcie_mty_rx_fifo[127-ww*8 - : 8];
   end
Endgenerate
```

上述公式等价于：

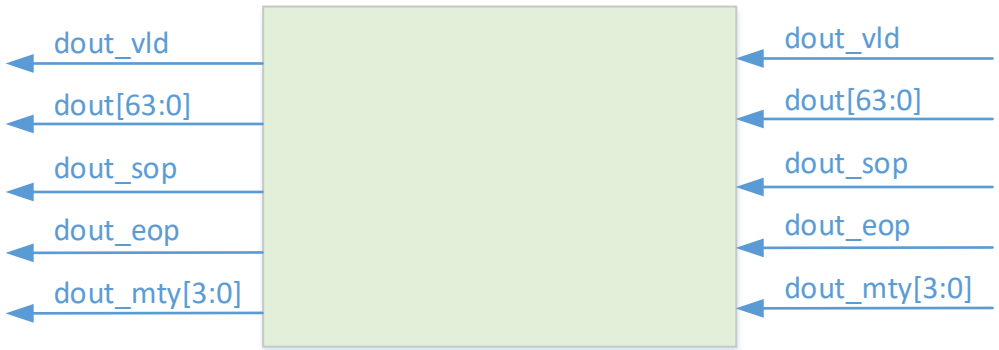
```
assign   pcie_mty_rx[0*8 +7 - :8] = pcie_mty_rx_fifo[127-0*8 - : 8];
assign   pcie_mty_rx[1*8 +7 - :8] = pcie_mty_rx_fifo[127-1*8 - : 8];
assign   pcie_mty_rx[2*8 +7 - :8] = pcie_mty_rx_fifo[127-2*8 - : 8];
assign   pcie_mty_rx[3*8 +7 - :8] = pcie_mty_rx_fifo[127-3*8 - : 8];
                                    .
assign   pcie_mty_rx[14*8 +7 - :8] = pcie_mty_rx_fifo[127-14*8 - : 8];
assign   pcie_mty_rx[15*8 +7 - :8] = pcie_mty_rx_fifo[127-15*8 - : 8];
// 解释   ：  [7 - : 8]
// 等价于：  [a - : b ]  == [a : a - b ] == [7 :0 ]    // 因为是从0 开始数的，  7 6 5 4 3 2 1 0
```

上述公式等价于：

```
assign   pcie_mty_rx[ 7  :  0] = pcie_mty_rx_fifo [127 : 120];
assign   pcie_mty_rx[15 :  8] = pcie_mty_rx_fifo [119 : 112];
assign   pcie_mty_rx[23 :16] = pcie_mty_rx_fifo [111 : 104];
assign   pcie_mty_rx[31 :24] = pcie_mty_rx_fifo [103 :  96];
```

上一页

模块名　　：mdy_pack_pick_by_head_guangqian
模块例化名：u_rx_cpack_analy
作用　　：判断数据头是不是16'hF8F8
　　光纤转存DDR3



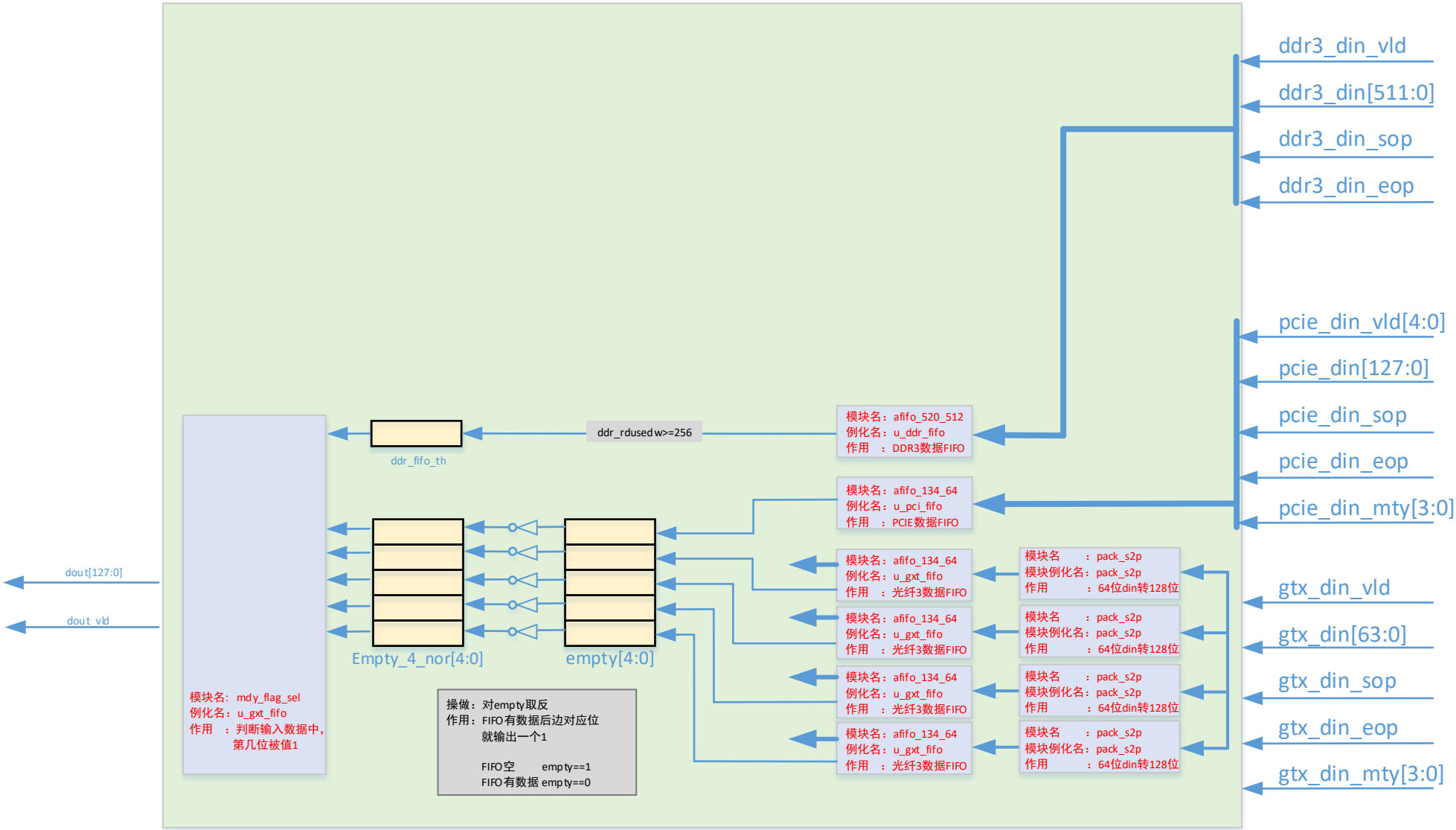

```
If( flag_add ==0 && din_sop && din_vld && din[63:48] == 16'hf8f8 )
    flag_add<=1;
else if( flag_add==1 && din_eop && din_vld )
   flag_add<=0;

...
   dout <=din;
   dout_sop <=din_sop;
   dout_eop <=din_eop;
   dout_vld <= dout_vld &&  flag_add;
...
```

框图

模块名：u_tx_sp

作用：6个FIFO选择一个输出到PCEI的TX

同时对不满128位的数据包做64位转128位

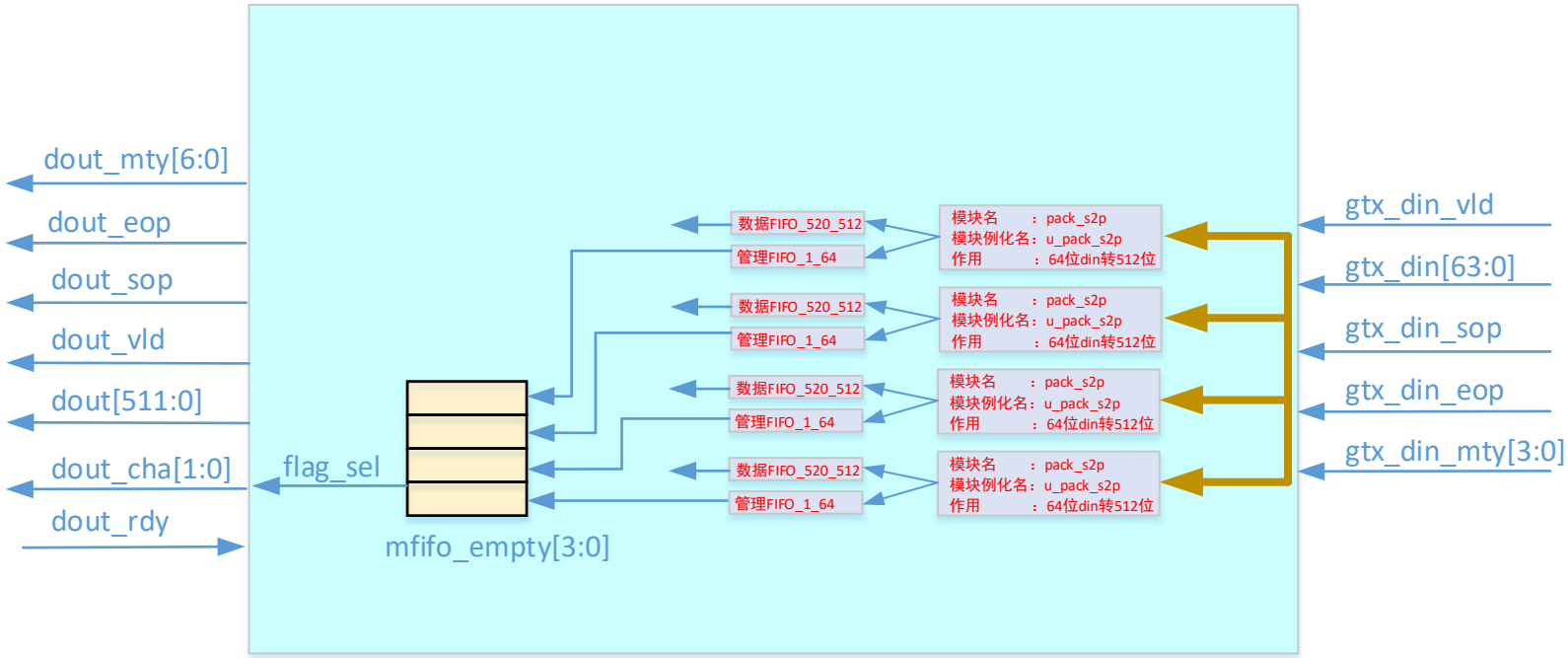


框图

```
//**********************
//  判断输入数据中，第几位被值1
//**********************/
//举例：输入数据例子：
        1.if(din==5'b00001)
        2.if(din==5'b00010)
        3.if(din==5'b00100)
        4.if(din==5'b01000)
        5.if(din==5'b10000)
//计算：先计算哪一位有效：
        sel[0 ]=din[0    ];
        sel[1 ]=sel[0 :0]==0 && din[1 ];
        sel[2 ]=sel[1 :0]==0 && din[2 ];
        sel[3 ]=sel[2 :0]==0 && din[3 ];
        sel[4 ]=sel[3 :0]==0 && din[4 ];
        sel[5 ]=sel[4 :0]==0 && din[5 ];
//计算并输出：判断并输出
        dout_bin[0] = sel[1 ] | sel[3 ] | sel[5 ] ;
        dout_bin[1] = sel[2 ] | sel[3 ] ;
        dout_bin[2] = sel[4 ] | sel[5 ] ;
 //参考：十进制表达法和二进制表达法
     5'd1  ==  3'b001
     5'd2  ==  3'b010
     5'd3  ==  3'b011
     5'd4  ==  3'b100
     5'd5  ==  3'b101
```

上一页

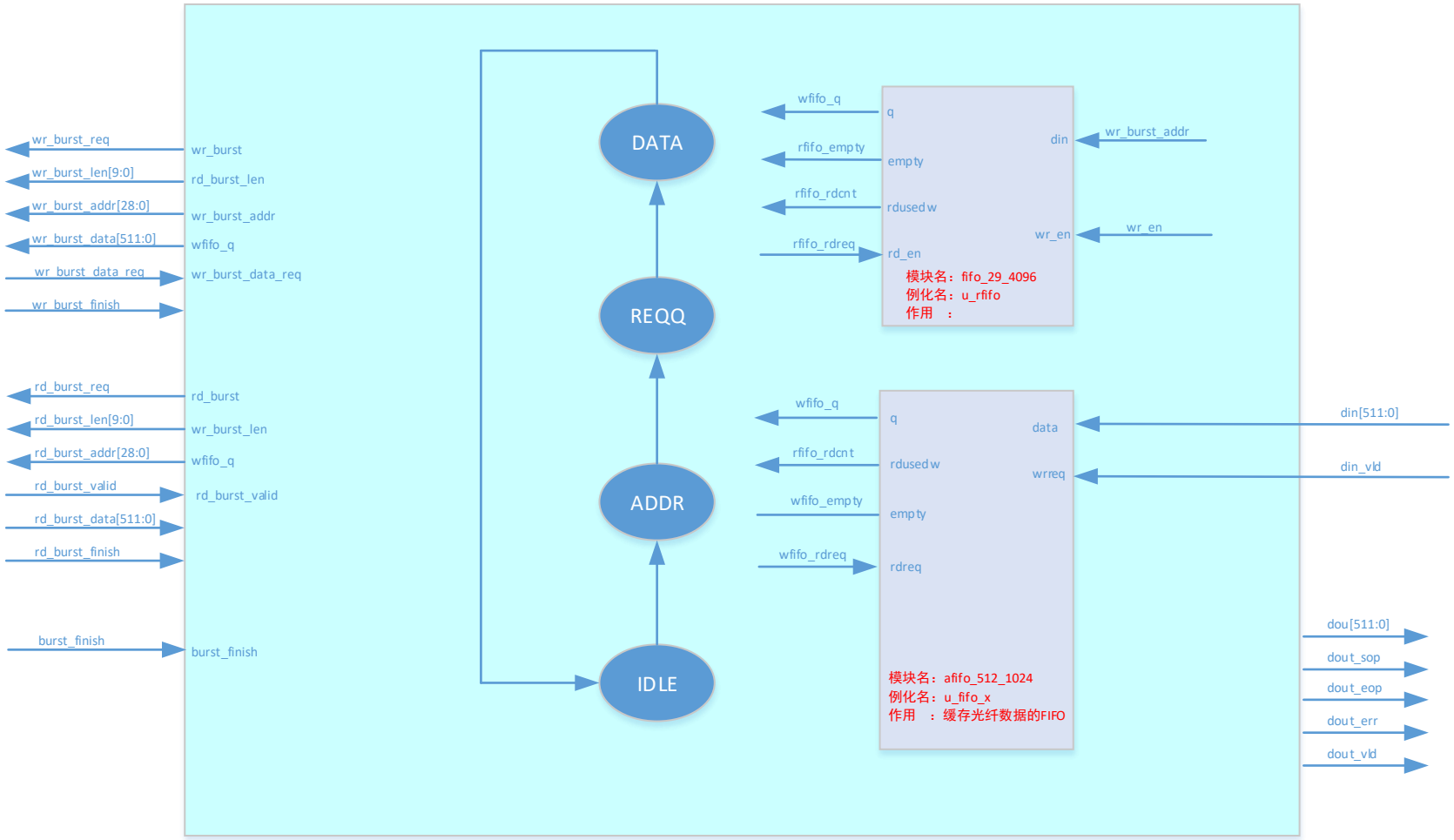
模块名　　：gtx2ddrSp
模块例化名：rx_sp
作用　　：四路光纤选择哪一路写入到DDR中



框图

模块名：ddr_mem_intf_guangxian
例化名：u_ddr_men_intf
作用　：把光纤过来的数据，缓存到FIFO中，
达到一定数量(DDR3_IP核的突发长度)。就把数据写入到DDR3中。

写达到设定数量后，开始读取DDR3中的数据。
给 PCIE发送过去



框图

命令总线

读总线

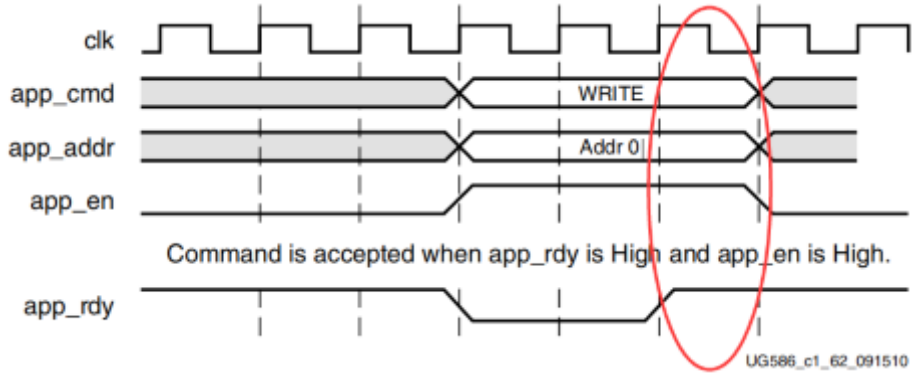
写

信号	方向	含义
app_addr	input	地址总线，1bit 地址=16bit 数据
app_cmd	input	命令总线，3'b000 代表写，3'b001 代表读
app_en	input	使能信号，高有效。 app_en 和 app_rdy 均有效时，app_cmd 和 app_addr 才有效
app_rdy	output	空闲信号，高有效。 app_en 和 app_rdy 均有效时，app_cmd 和 app_addr 才有效
app_hi_pri	input	优先级信号，高有效，用于提高本次请求的优先级
app_rd_data	output	读数据总线
app_rd_data_end	output	1 bit 最后一个读数据的标志，该信号高有效时，代表对应的 app_rd_data 为当前写的最后一个数据
app_rd_data_valid	output	读数据有效标志
app_sz	input	保留信号，置 0
app_wdf_data	input	128 bit 写数据总线，128/16=8，因此地址 app_addr 递增 8 bit
app_wdf_end	input	1 bit 最后一个写数据的标志，该信号高有效时，代表对应的 app_wdf_data 为当前写的最后一个数据
app_wdf_mask	input	16 bit 写数据掩码，为 1 则掩盖掉，1 bit 对应 app_wdf_data 的一个字节
app_wdf_rdy	output	1 bit 写数据空闲信号，该信号高有效，且 app_wdf_rdy 也有效时，IP 核才可以接收到用户端发送的 app_wdf_data
app_wdf_wren	input	1 bit 写数据有效标志，该信号高有效，且 app_wdf_rdy 也有效时，IP 核才可以接收到用户端发送的 app_wdf_data
app_correct_en_i	input	纠正信号，高有效，断言时纠正单比特数据。只有在 GUI 中启用 ECC 时，此输入才有效。此信号总是置 1。
app_sr_req	input	保留信号，置 0
app_sr_active	output	保留信号
app_ref_req	input	请求向 DRAM 发出刷新命令，高有效
app_ref_ack	output	将所请求的刷新命令发送到 PHY 接口，高有效
app_zq_req	input	向 DRAM 发出 ZQ 校准命令，高有效
app_zq_ack	output	向 PHY 接口发送所请求的 ZQ 校准命令，高有效
ui_clk	output	时钟信号，必须是 DRAM 时钟的 1/2 或 1/4
init_calib_complete	output	初始化标志信号，高有效
app_ecc_multiple_err	output	此信号仅显示到 MEMC_UI_TOP 模块，只应在启用 ECC 时使用。
ui_clk_sync_rst	output	复位信号，高有效

上一页
下一页

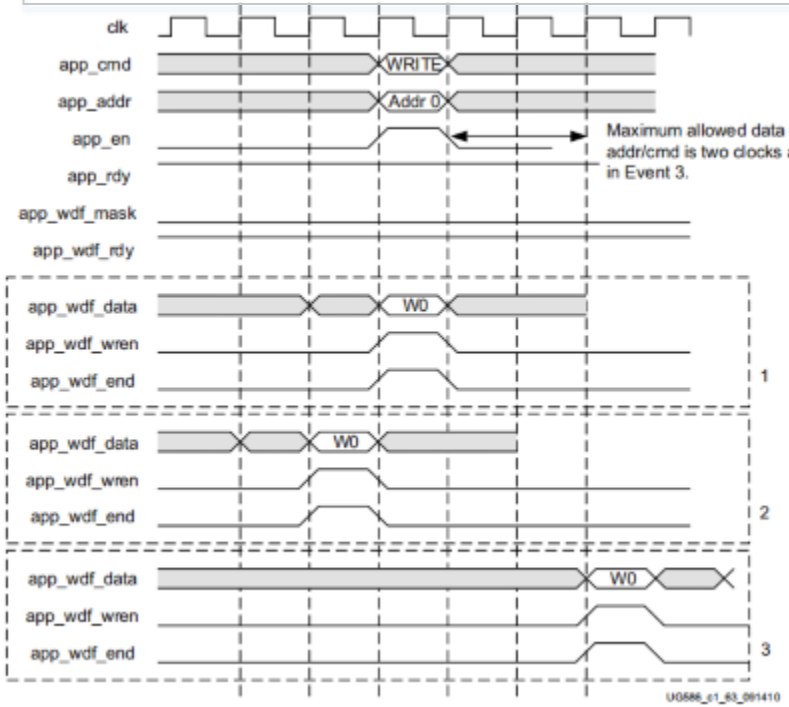
首页

命令总线



写总线

写总线（表格黄色部分），其时序图如下所示。共有 3 种传输模式。
模式 1 指的是命令和数据同时发送到 IP 核，模式 2 指的是数据提前于命令发送到 IP 核，
模式 3 指的是数据落后于命令发送到 IP 核。模式 1 和 2 均可稳定传输，
而模式 3 必须满足一个条件，即数据落后命令的时间不能超过两个时钟周期。
一般来说，模式 2 比较常用，时序设计相对容易



读总线

读总线（表格黄色部分），也分为两种速率，4:1 和 2:1。

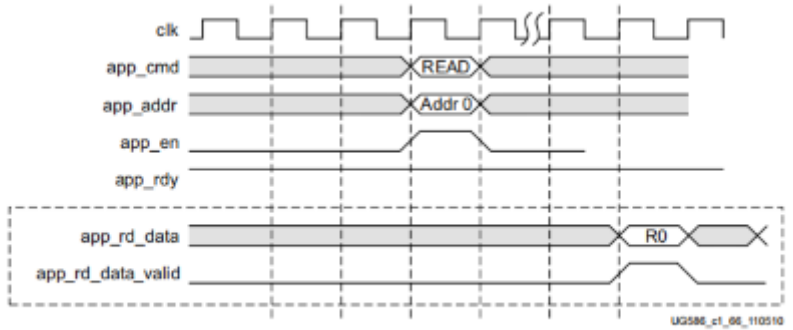


Figure 1-81: **4:1 Mode UI Interface Read Timing Diagram (Memory Burst Type = BL8)**

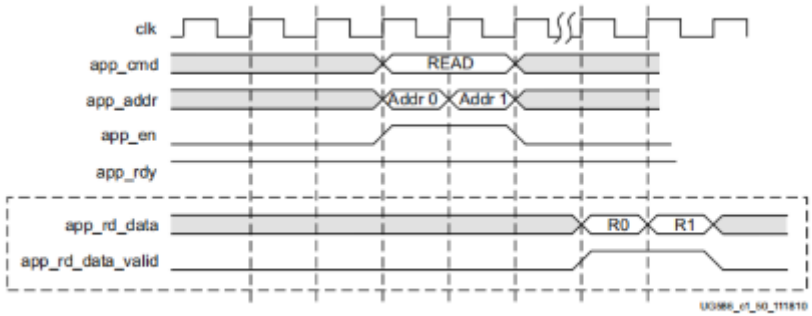
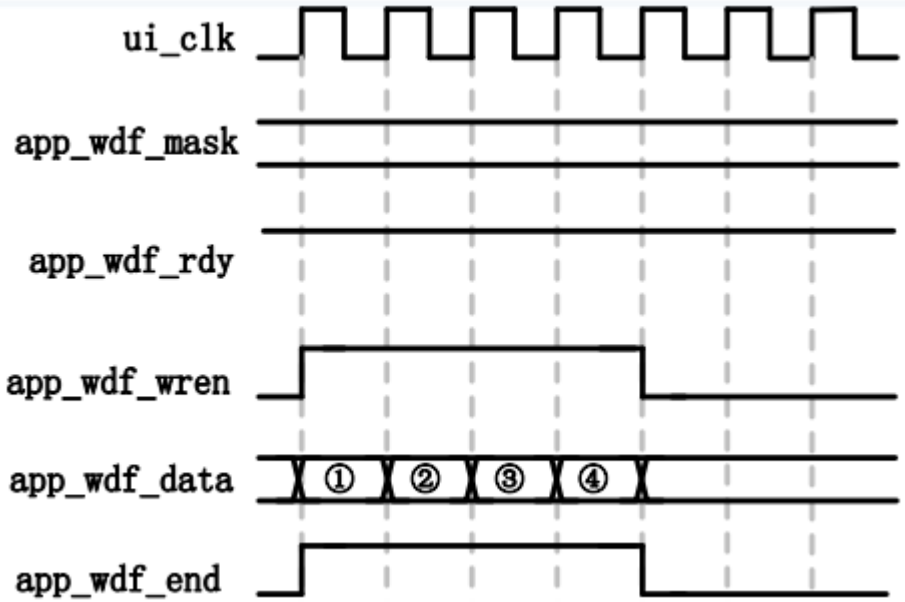
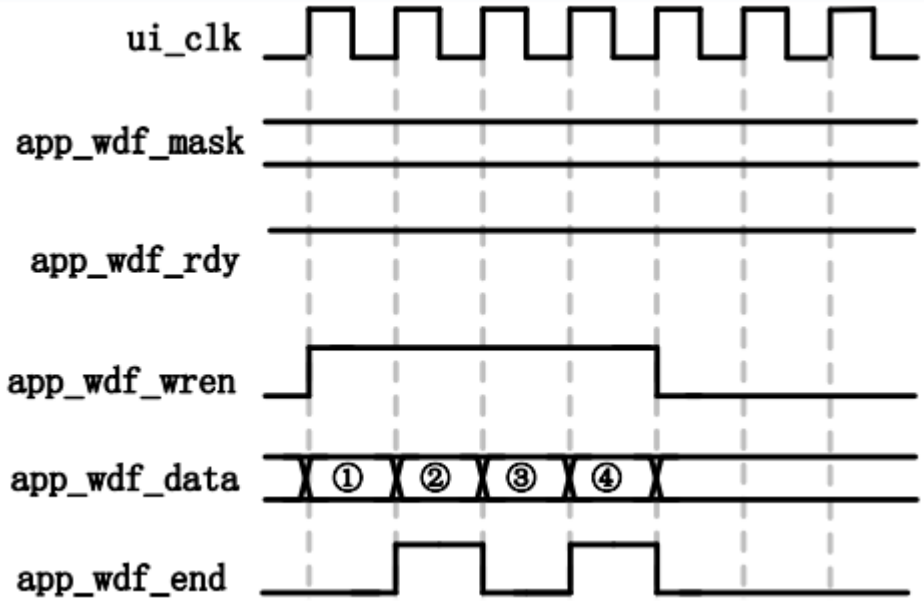


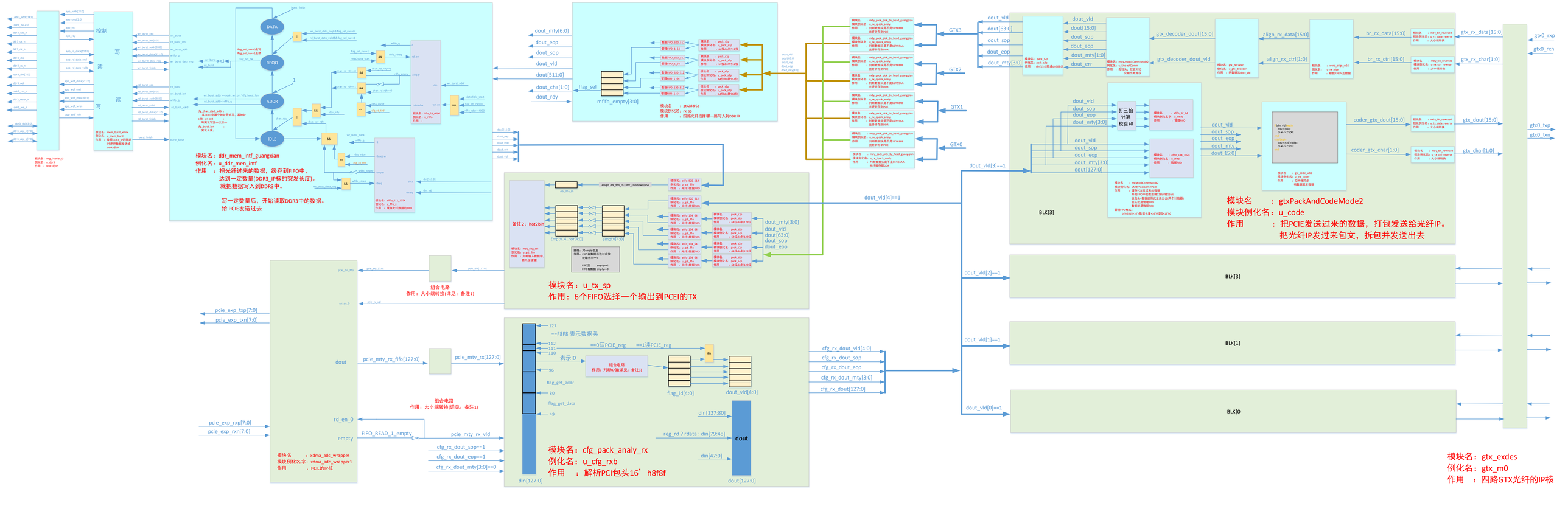
Figure 1-82: **2:1 Mode UI Interface Read Timing Diagram (Memory Burst Type = BL4 or BL8)**

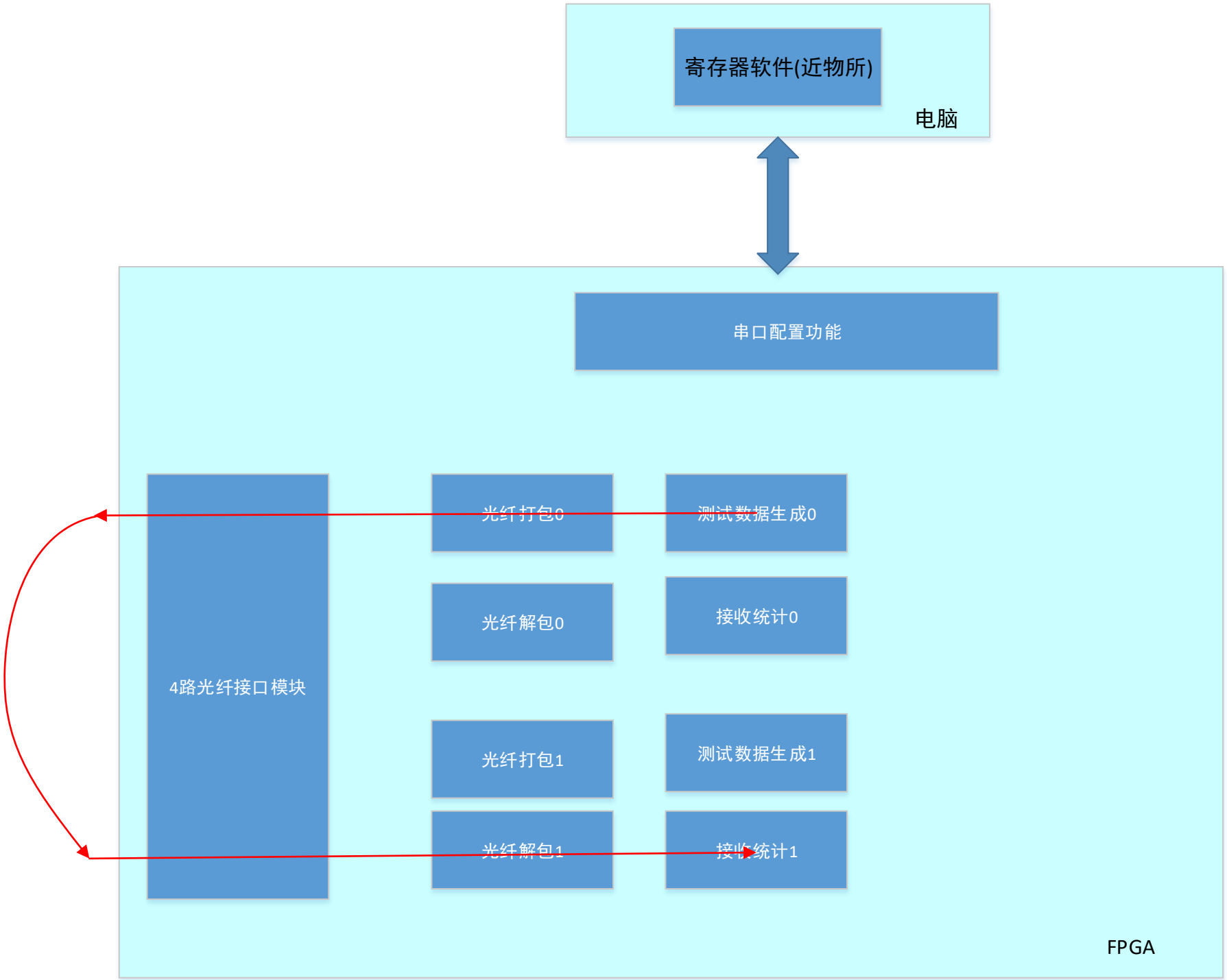
关于 app_wdf_end 信号，该信号表示：当前突发写的最后一个数据。我们此次 DDR3 IP 核的突发长度为 8 ，1 个的长度是 16bit。
本次 DDR3 IP 核调取时，我们选取的 “物理层 - 用户端” 的速率为 4：1，每次发送的有效数据为 128 bit，app_wdf_data 本就是128 bit的，
因此 1 次突发写就完成了数据的写入，app_wdf_end 和 app_wdf_en 时序上同步了。

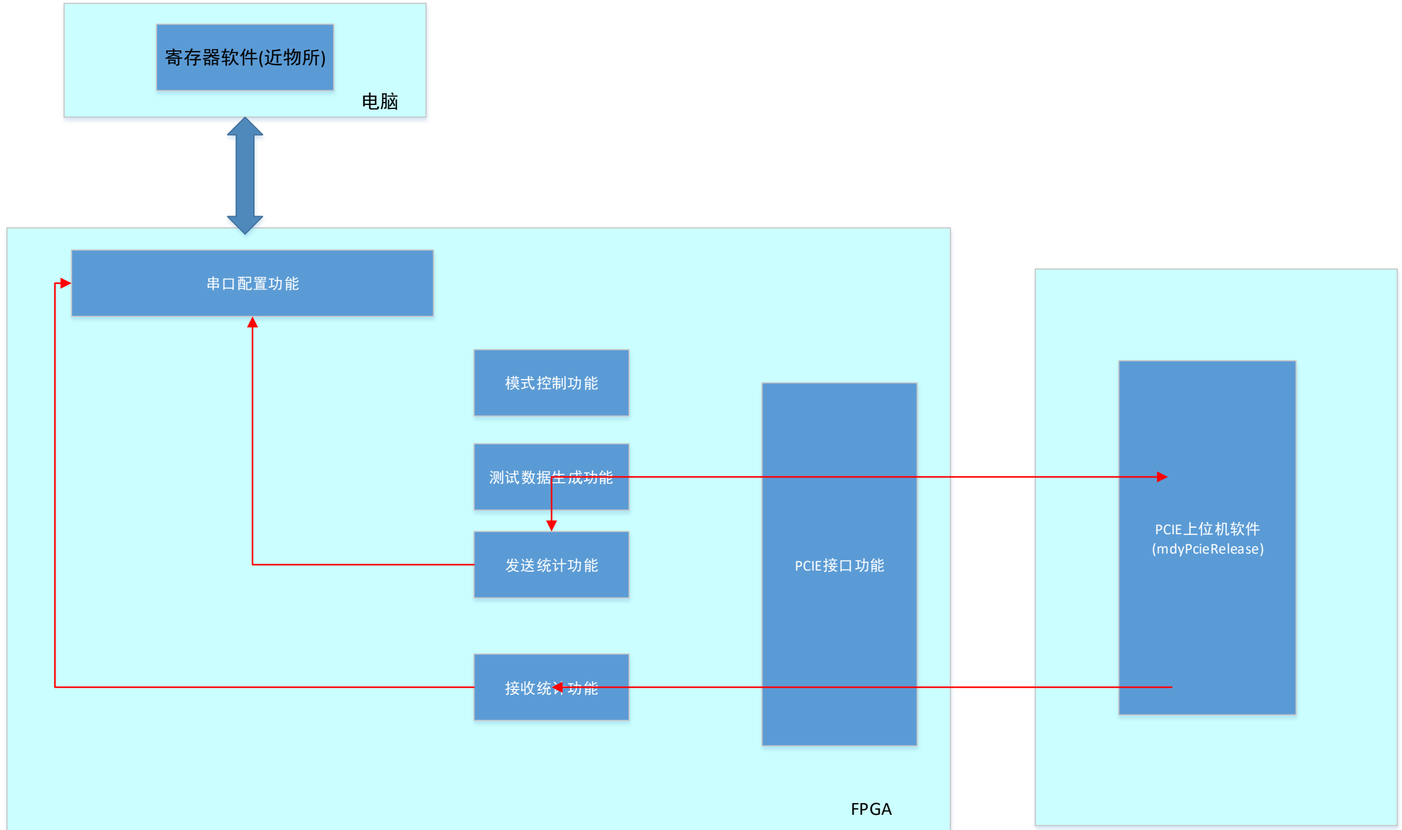


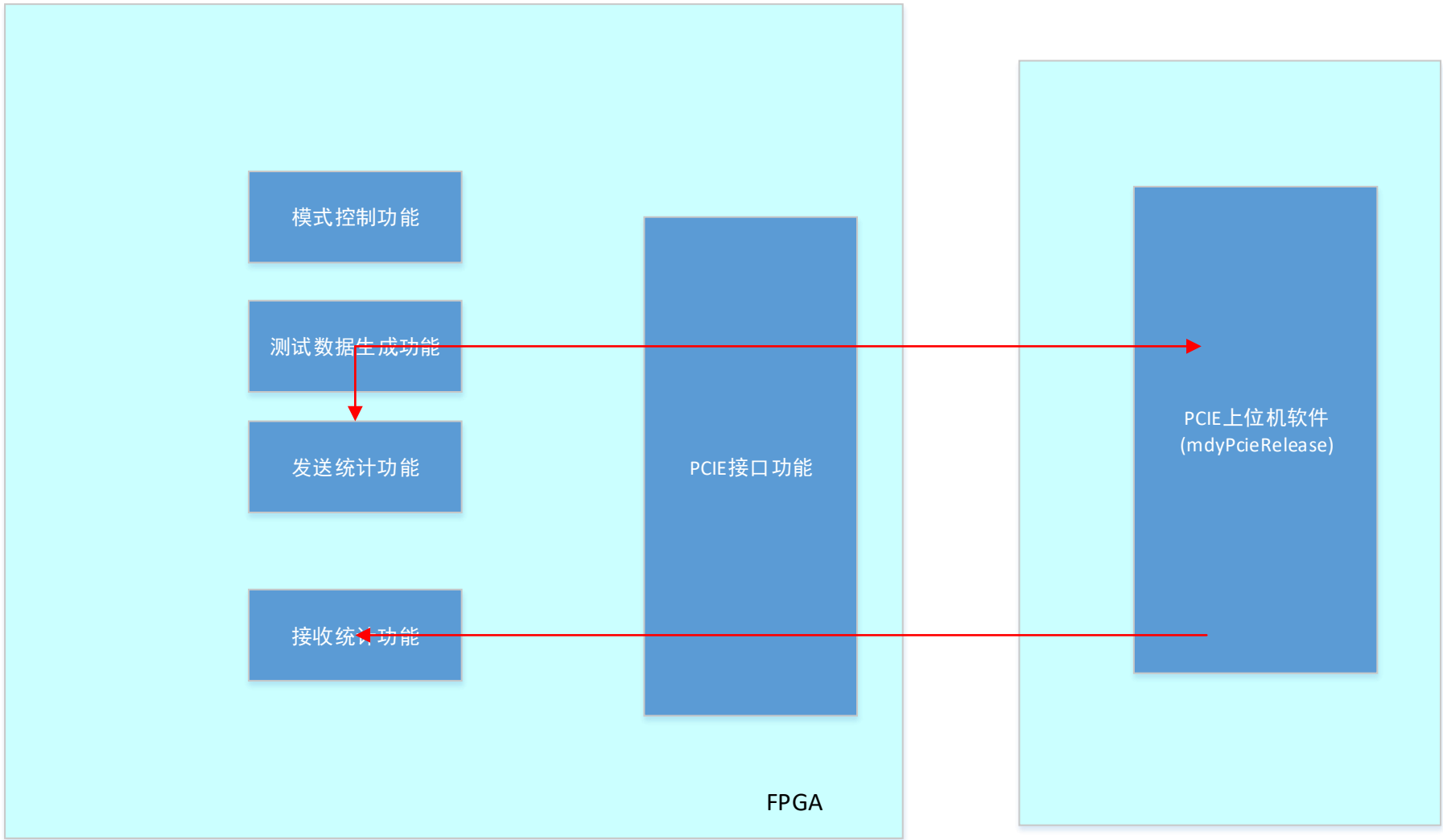
而如果选取的 “物理层 - 用户端” 的速率为 2：1，那每次发送的有效数据为 64 bit，
app_wdf_data 本是128 bit的，因此需要 2 次突发才能完成数据的写入，app_wdf_end 就看得更清楚了

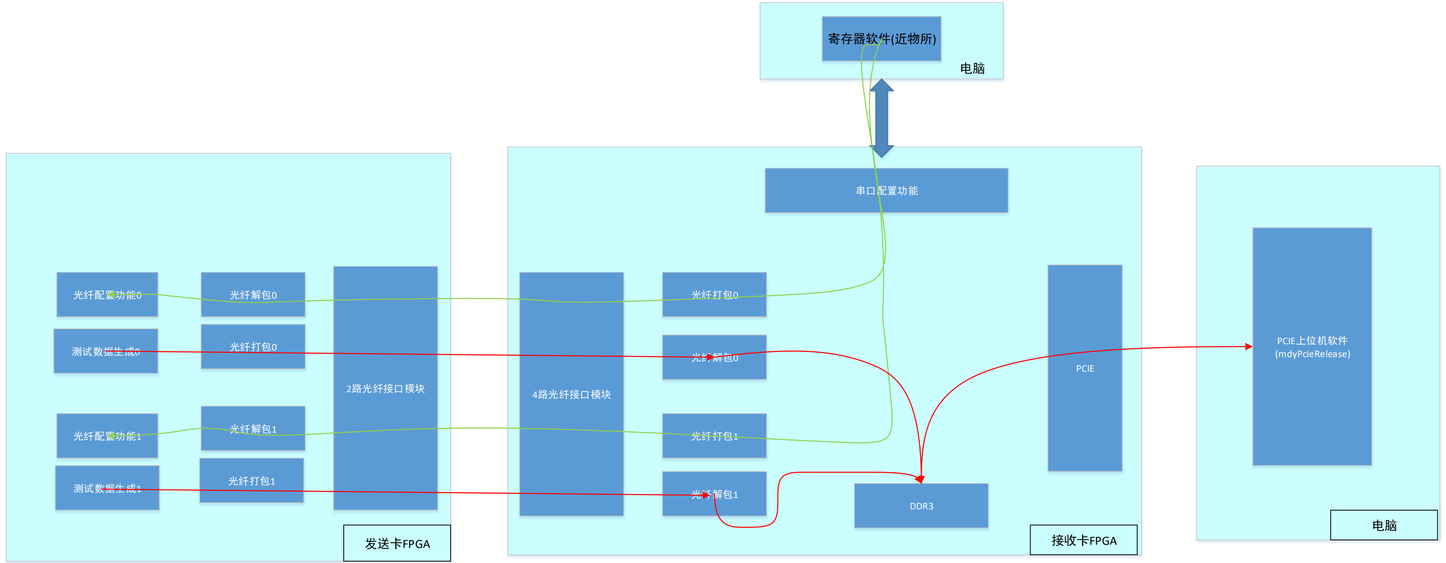



模块名：ddr_mem_intf_guangxian
例化名：u_ddr_men_intf
作用 ：把光纤过来的数据，缓存到FIFO中，
达到一定数量(DDR3_IP核的突发长度)。
就把数据写入到DDR3中。
写一定数量后，开始读取DDR3中的数据。
给 PCIE发送过去
模块名 ：u_tx_sp
作用：6个FIFO选择一个输出到PCEI的TX
模块名：cfg_pack_analy_rx
例化名：u_cfg_rxb
作用 ：解析PCI包头16’h8f8f
模块名 ：gtxPackAndCodeMode2
模块例化名：u_code
作用 ：把PCIE发送过来的数据，打包发送给光纤IP。
把光纤IP发过来包文，拆包并发送出去
模块名：gtx_exdes
例化名：gtx_m0
作用 ：四路GTX光纤的IP核


寄存器软件(近物所)
电脑
串口配置功能
4路光纤接口模块
光纤打包0
测试数据生成0
光纤解包0
接收统计0
光纤打包1
测试数据生成1
光纤解包1
接收统计1
FPGA


寄存器软件(近物所)
电脑
串口配置功能
模式控制功能
测试数据生成功能
发送统计功能
接收统计功能
PCIE接口功能
FPGA
PCIE上位机软件
(mdyPcieRelease)


模式控制功能
测试数据生成功能
发送统计功能
接收统计功能
PCIE接口功能
FPGA
PCIE上位机软件
(mdyPcieRelease)


寄存器软件(近物所)
电脑
串口配置功能
光纤配置功能0
光纤解包0
测试数据生成0
光纤打包0
光纤配置功能1
光纤解包1
测试数据生成1
光纤打包1
2路光纤接口模块
发送卡FPGA
4路光纤接口模块
光纤打包0
光纤解包0
光纤打包1
光纤解包1
DDR3
PCIE
接收卡FPGA
PCIE上位机软件
(mdyPcieRelease)
电脑