基于FPGA的接口转换——MiPi转HDMI

主讲人：Nill

课程的意义不在于课程结束后，你认为我说的都是对的，我们的最大动力是：通过这门课程的学习，你能证明，我是对的或者我是错的。

——深圳市我是你的眼有限公司 ——

专业的FPGA、无线通信方案商

微信公众号：MYMINIEYE
答疑邮箱：support@myminieye.com
网址：www.myminieye.com
淘宝店铺：小眼睛半导体

关注&交流

- 微信公众号：
- 讨论群：

微信讨论群二维码

QQ讨论群二维码

课程目标：

- 解析CSI转HDMI的设计流程

  1）从DPHY输出并行流中提取有效数据；

  2）CSI数据流结构；

  3）视频流信号缓冲；

  4）TMDS编码串行输出；

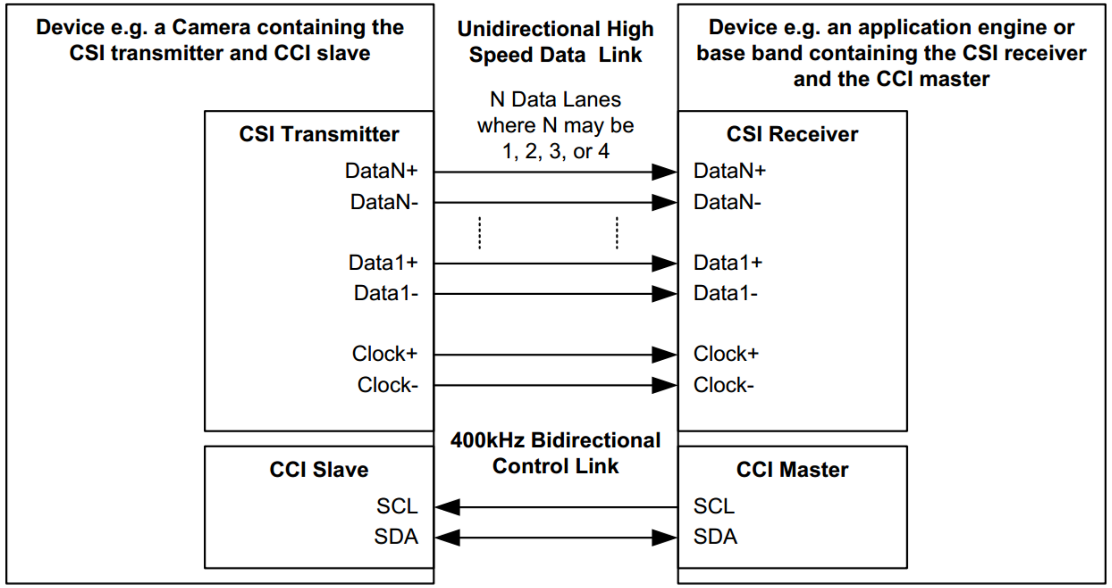
目 录

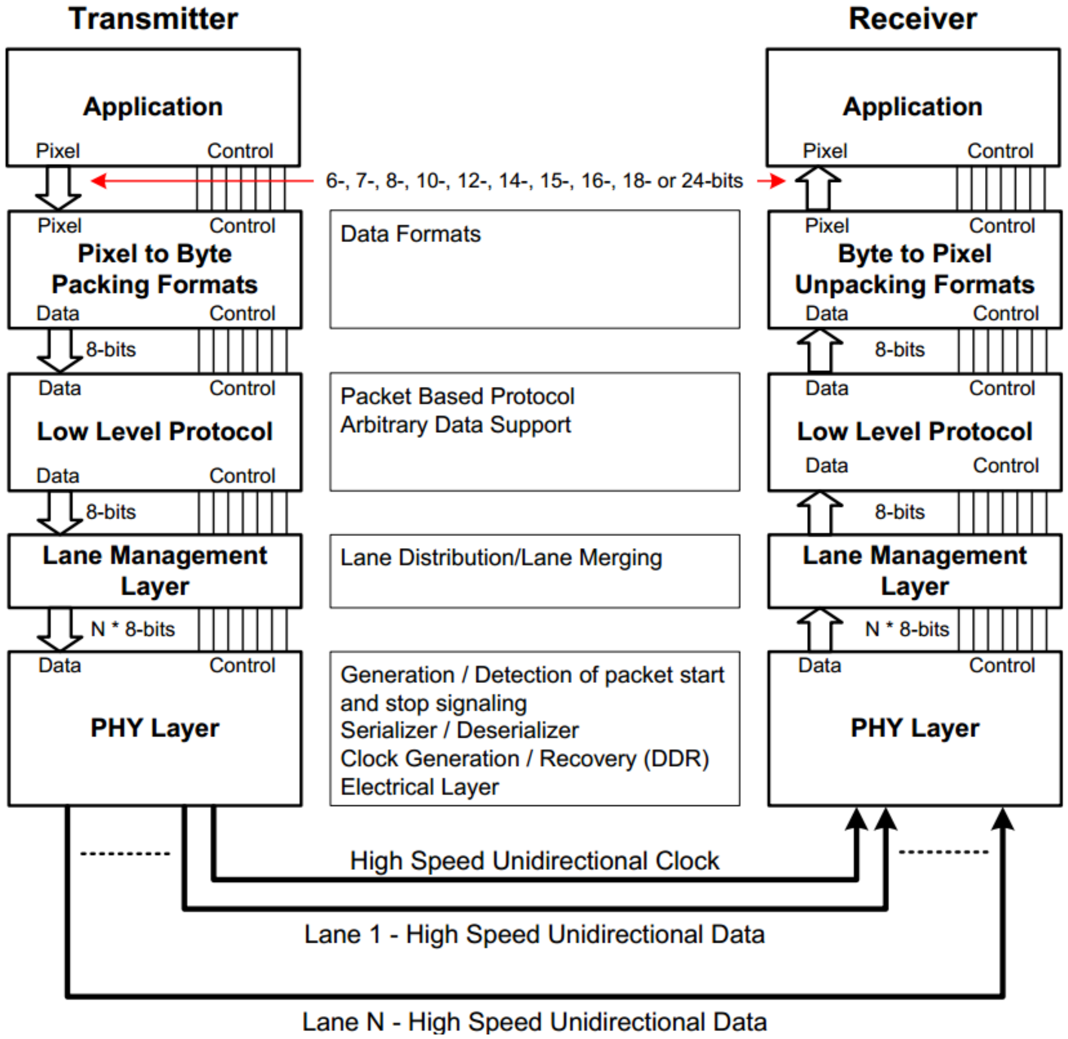
CONTENTS

C O N T E N T S

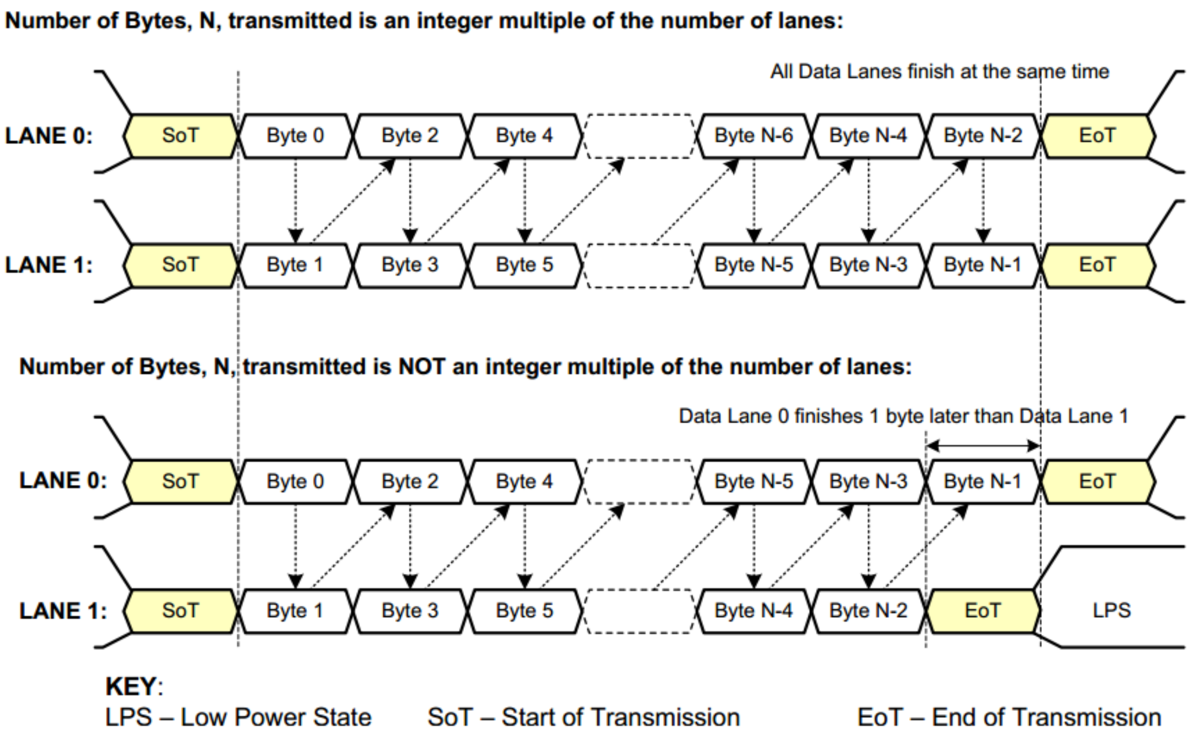
01 CSI2接口

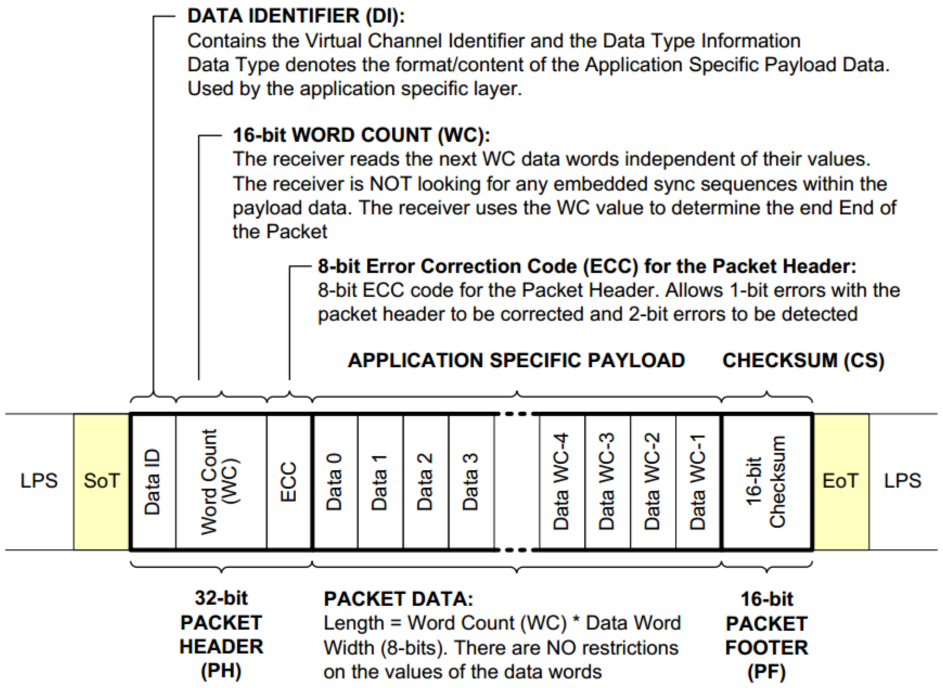
CSI连接链路及接收层级定义

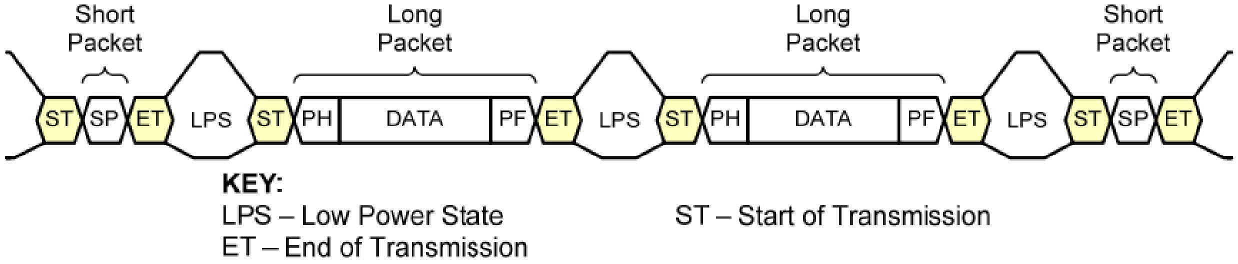


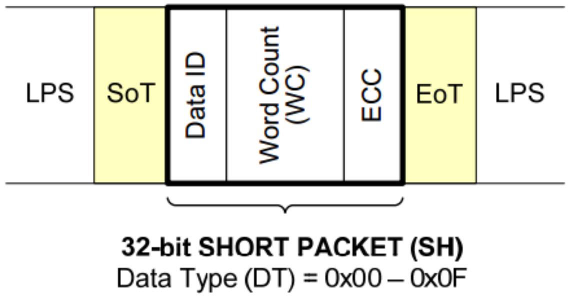


Lane control capture









DATA IDENTIFIER

Table 3 Data Type Classes

Data Type	Description
0x00 to 0x07	Synchronization Short Packet Data Types
0x08 to 0x0F	Generic Short Packet Data Types
0x10 to 0x17	Generic Long Packet Data Types
0x18 to 0x1F	YUV Data
0x20 to 0x27	RGB Data
0x28 to 0x2F	RAW Data
0x30 to 0x37	User Defined Byte-based Data
0x38 to 0x3F	Reserved

P74
P84
P88
P96

Data Type	Description
0x18	YUV420 8-bit
0x19	YUV420 10-bit
0x1A	Legacy YUV420 8-bit
0x1B	Reserved
0x1C	YUV420 8-bit (Chroma Shifted Pixel Sampling)
0x1D	YUV420 10-bit (Chroma Shifted Pixel Sampling)
0x1E	YUV422 8-bit
0x1F	YUV422 10-bit

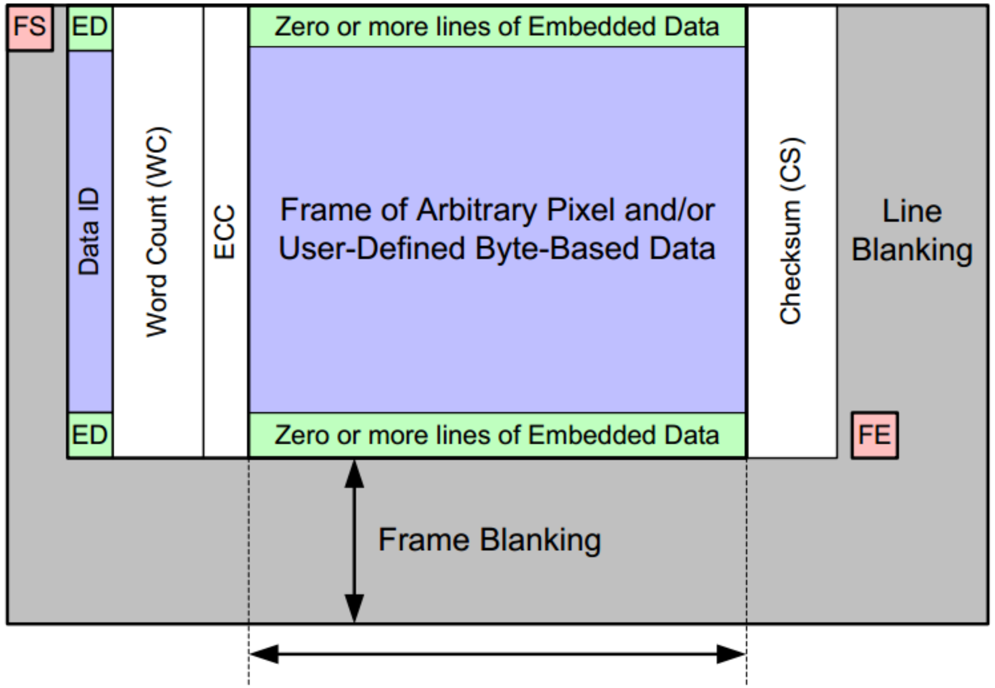
Data Type	Description
0x20	RGB444
0x21	RGB555
0x22	RGB565
0x23	RGB666
0x24	RGB888
0x25	Reserved
0x26	Reserved
0x27	Reserved

Table 6 Synchronization Short Packet Data Type Codes

Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Optional)
0x03	Line End Code (Optional)
0x04 to 0x07	Reserved

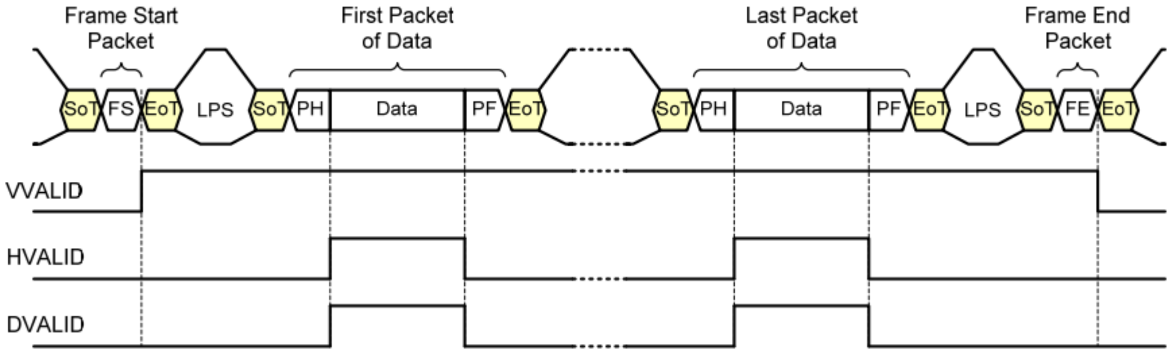
Data Type	Description
0x10	Null
0x11	Blanking Data
0x12	Embedded 8-bit non Image Data
0x13	Reserved
0x14	Reserved
0x15	Reserved
0x16	Reserved
0x17	Reserved

图像帧格式



Payload Data per packet must be a multiple of 8-bits

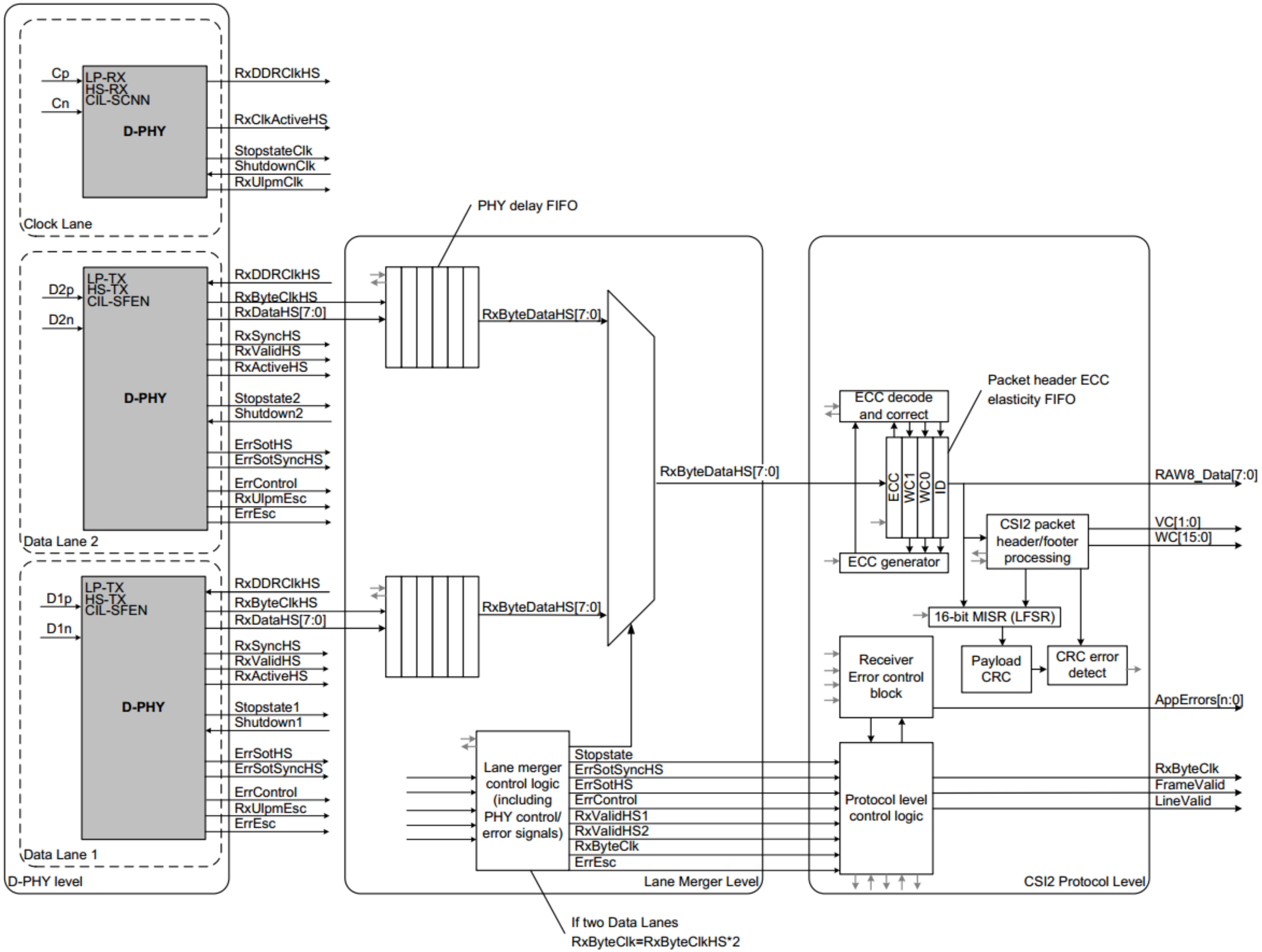
KEY:
LPS – Low Power State
ECC – Error Correction Code
FS – Frame Start
LS – Line Start
DI – Data Identifier
CS – Checksum
FE – Frame End
LE – Line End
WC – Word Count
ED – Embedded Data



KEY:
SoT – Start of Transmission
PH – Packet Header
FS – Frame Start
LS – Line Start
EoT – End of Transmission
PF – Packet Footer
FE – Frame End
LE – Line End
LPS – Low Power State

Figure 45 Multiple Packet Example

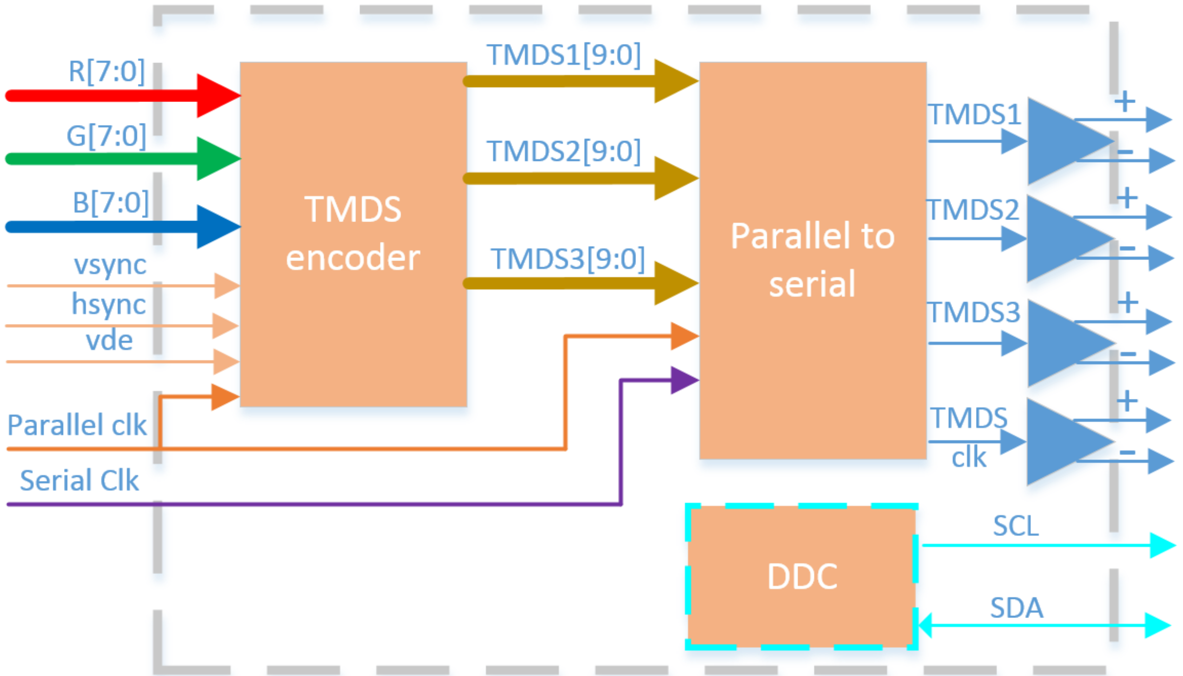
CSI2 接收框图

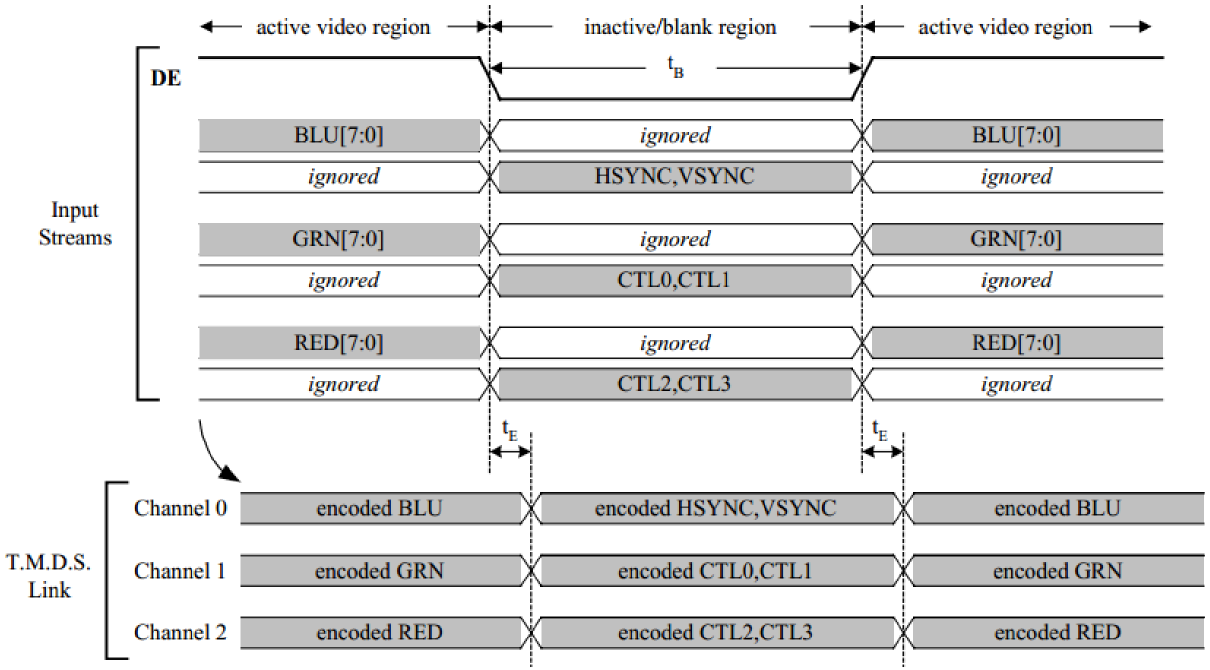


Cp
Cn
LP-RX
HS-RX
CIL-SCNN
D-PHY
RxDDRClkHS
RxClkActiveHS
StopstateClk
ShutdownClk
RxUlpmClk
Clock Lane
D2p
D2n
LP-TX
HS-TX
CIL-SFEN
D-PHY
RxDDRClkHS
RxByteClkHS
RxDataHS[7:0]
RxSyncHS
RxValidHS
RxActiveHS
Stopstate2
Shutdown2
ErrSotHS
ErrSotSyncHS
ErrControl
RxUlpmEsc
ErrEsc
Data Lane 2
D1p
D1n
LP-TX
HS-TX
CIL-SFEN
D-PHY
RxDDRClkHS
RxByteClkHS
RxDataHS[7:0]
RxSyncHS
RxValidHS
RxActiveHS
Stopstate1
Shutdown1
ErrSotHS
ErrSotSyncHS
ErrControl
RxUlpmEsc
ErrEsc
Data Lane 1
D-PHY level
PHY delay FIFO
RxByteDataHS[7:0]
RxByteDataHS[7:0]
RxByteDataHS[7:0]
Lane merger control logic (including PHY control/ error signals)
Stopstate
ErrSotSyncHS
ErrSotHS
ErrControl
RxValidHS1
RxValidHS2
RxByteClk
ErrEsc
Lane Merger Level
If two Data Lanes
RxByteClk=RxByteClkHS*2
ECC decode and correct
Packet header ECC elasticity FIFO
ECC
WC1
WC0
ID
ECC generator
RAW8_Data[7:0]
CSI2 packet header/footer processing
VC[1:0]
WC[15:0]
16-bit MISR (LFSR)
Receiver Error control block
Payload CRC
CRC error detect
AppErrors[n:0]
Protocol level control logic
RxByteClk
FrameValid
LineValid
CSI2 Protocol Level

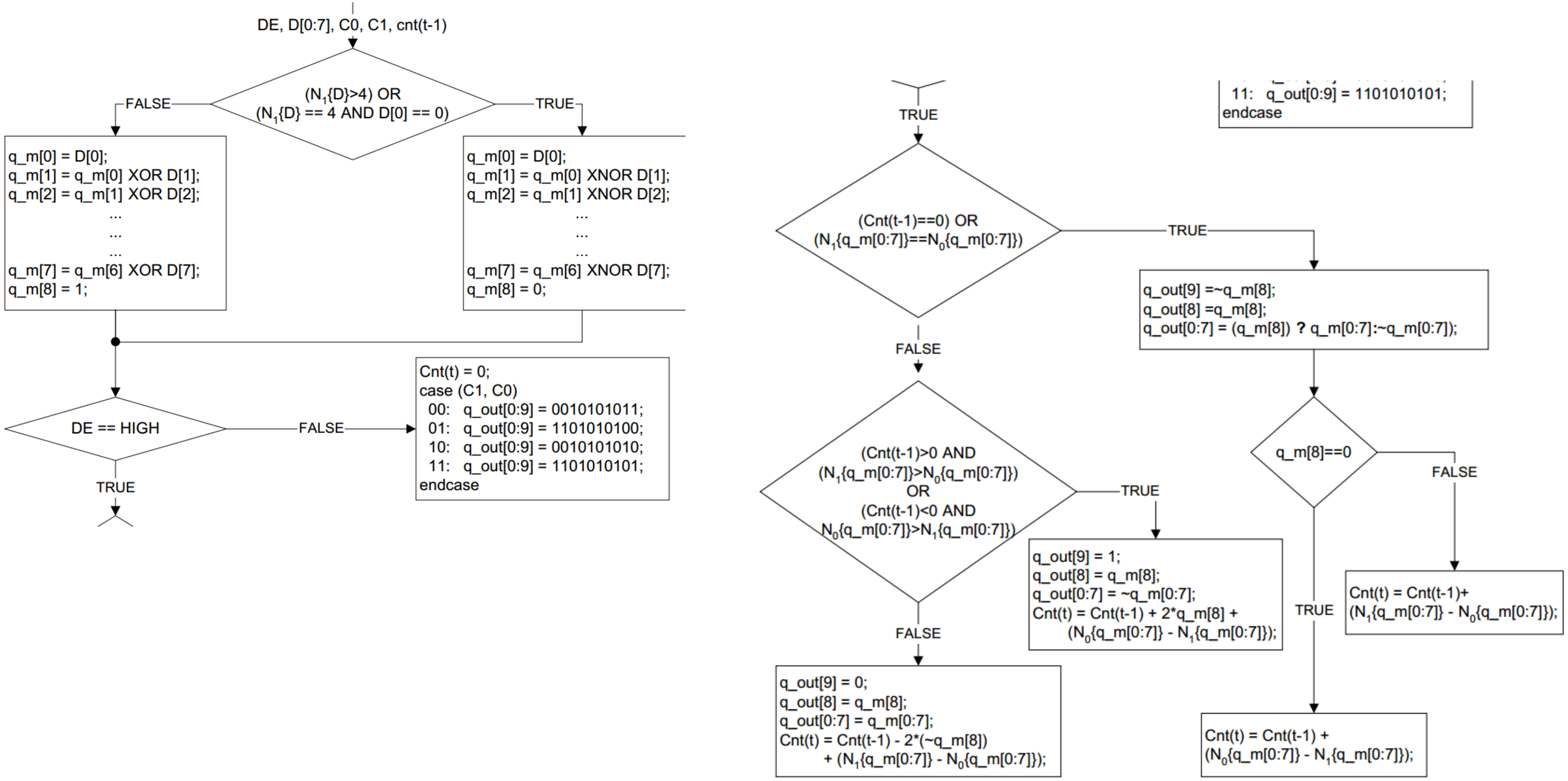
02 HDMI输出接口

RGB2DVI


R[7:0]
G[7:0]
B[7:0]
vsync
hsync
vde
Parallel clk
Serial Clk
TMDS encoder
TMDS1[9:0]
TMDS2[9:0]
TMDS3[9:0]
Parallel to serial
TMDS1
TMDS2
TMDS3
TMDS clk
+
-
DDC
SCL
SDA


active video region
inactive/blank region
active video region
DE
t_B
Input Streams
BLU[7:0]
ignored
ignored
HSYNC,VSYNC
GRN[7:0]
CTL0,CTL1
RED[7:0]
CTL2,CTL3
t_E
T.M.D.S. Link
Channel 0
Channel 1
Channel 2
encoded BLU
encoded HSYNC,VSYNC
encoded GRN
encoded CTL0,CTL1
encoded RED
encoded CTL2,CTL3

TMD编码流程


DE, D[0:7], C0, C1, cnt(t-1)
(N_1{D}>4) OR (N_1{D} == 4 AND D[0] == 0)
FALSE
TRUE
q_m[0] = D[0];
q_m[1] = q_m[0] XOR D[1];
q_m[2] = q_m[1] XOR D[2];
…
…
…
q_m[7] = q_m[6] XOR D[7];
q_m[8] = 1;
q_m[0] = D[0];
q_m[1] = q_m[0] XNOR D[1];
q_m[2] = q_m[1] XNOR D[2];
…
…
…
q_m[7] = q_m[6] XNOR D[7];
q_m[8] = 0;
DE == HIGH
FALSE
TRUE
Cnt(t) = 0;
case (C1, C0)
00: q_out[0:9] = 0010101011;
01: q_out[0:9] = 1101010100;
10: q_out[0:9] = 0010101010;
11: q_out[0:9] = 1101010101;
endcase
TRUE
11: q_out[0:9] = 1101010101;
endcase
(Cnt(t-1)==0) OR
(N_1{q_m[0:7]}==N_0{q_m[0:7]})
TRUE
FALSE
q_out[9] =~q_m[8];
q_out[8] =q_m[8];
q_out[0:7] = (q_m[8]) ? q_m[0:7]:~q_m[0:7]);
(Cnt(t-1)>0 AND
(N_1{q_m[0:7]}>N_0{q_m[0:7]})
OR
(Cnt(t-1)<0 AND
N_0{q_m[0:7]}>N_1{q_m[0:7]})
TRUE
FALSE
q_m[8]==0
FALSE
TRUE
q_out[9] = 1;
q_out[8] = q_m[8];
q_out[0:7] = ~q_m[0:7];
Cnt(t) = Cnt(t-1) + 2*q_m[8] +
(N_0{q_m[0:7]} - N_1{q_m[0:7]});
Cnt(t) = Cnt(t-1)+
(N_1{q_m[0:7]} - N_0{q_m[0:7]});
q_out[9] = 0;
q_out[8] = q_m[8];
q_out[0:7] = q_m[0:7];
Cnt(t) = Cnt(t-1) - 2*(~q_m[8])
+ (N_1{q_m[0:7]} - N_0{q_m[0:7]});
Cnt(t) = Cnt(t-1) +
(N_0{q_m[0:7]} - N_1{q_m[0:7]});

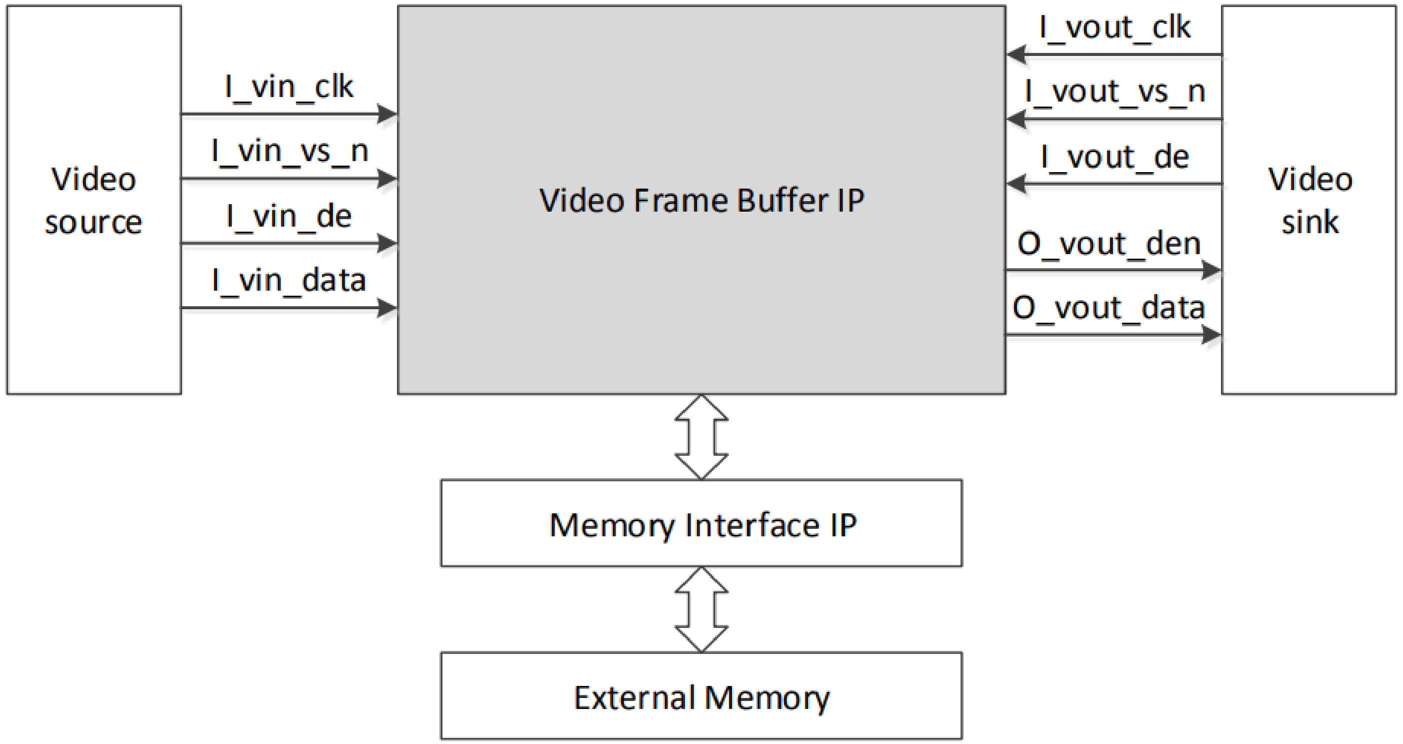
03

帧缓冲与系统框图

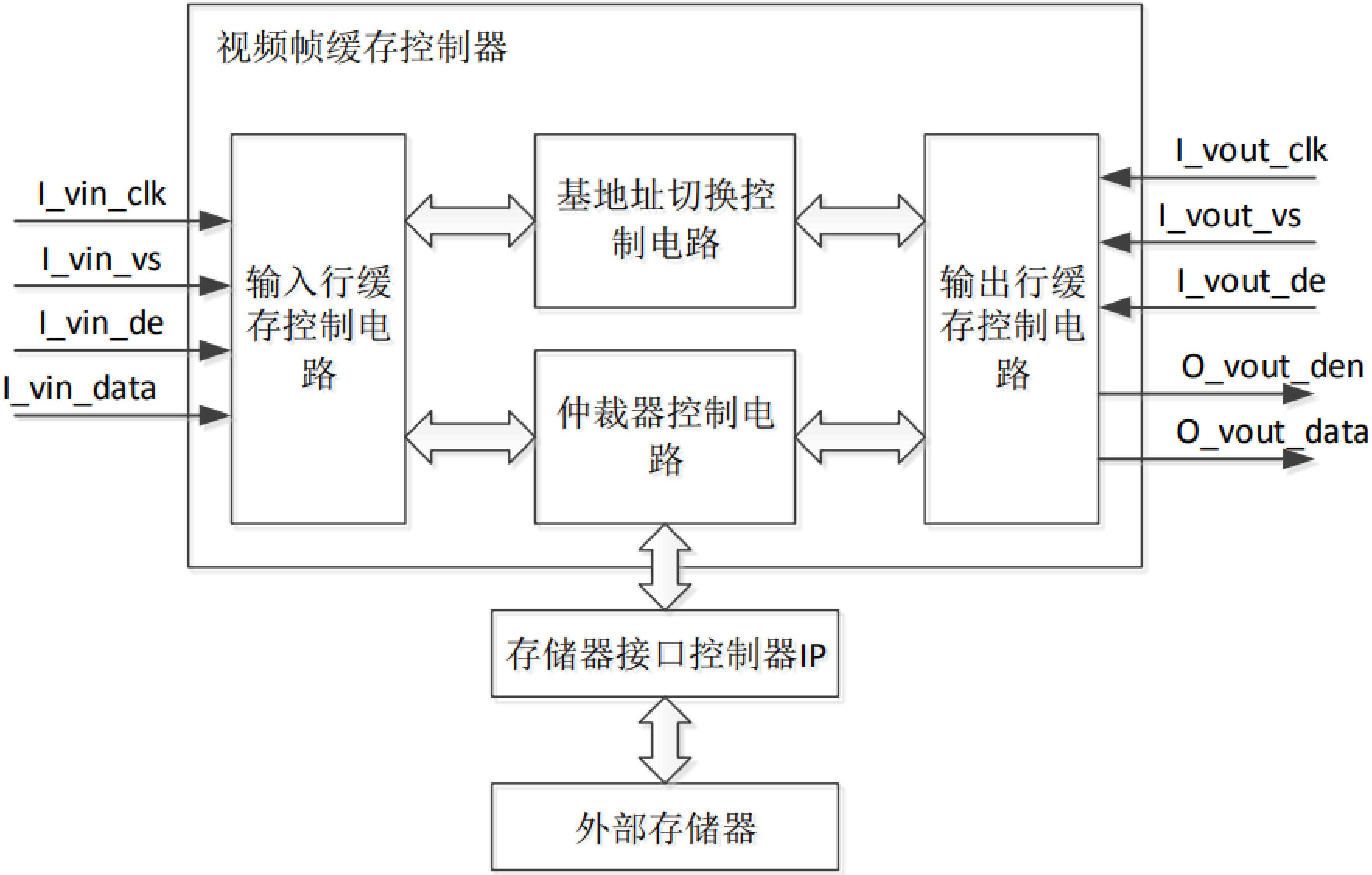
帧缓冲

1、MiPi接口输出数据率和MiPi接口跑的数据率相关，与vesa等显示协议定义的像素时钟的频率没有直接联系。
MiPi与HDMI之间时钟域不一致；

2、从MiPi数据流结构可得知解析出的数据是以帧为单位，提取相应每行的有效数据；
在HDMI输出需要有显示标准的行场同步信号，故接口转换中需要重建显示同步时序；

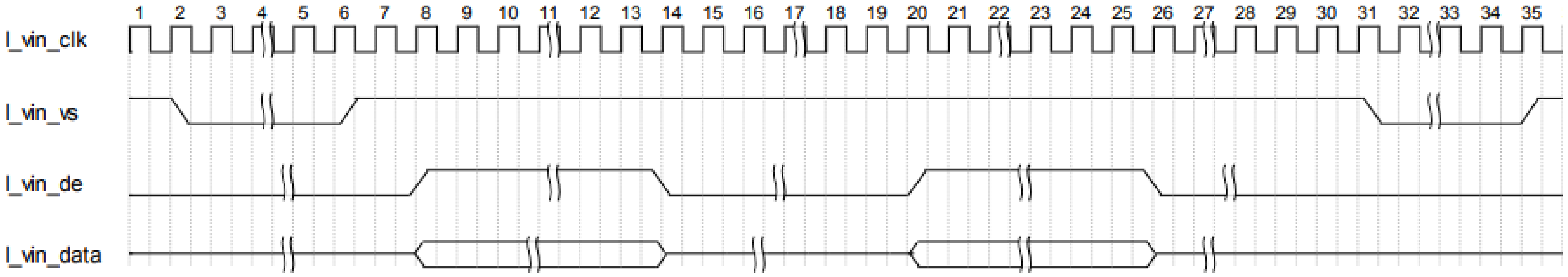


帧缓冲功能结构

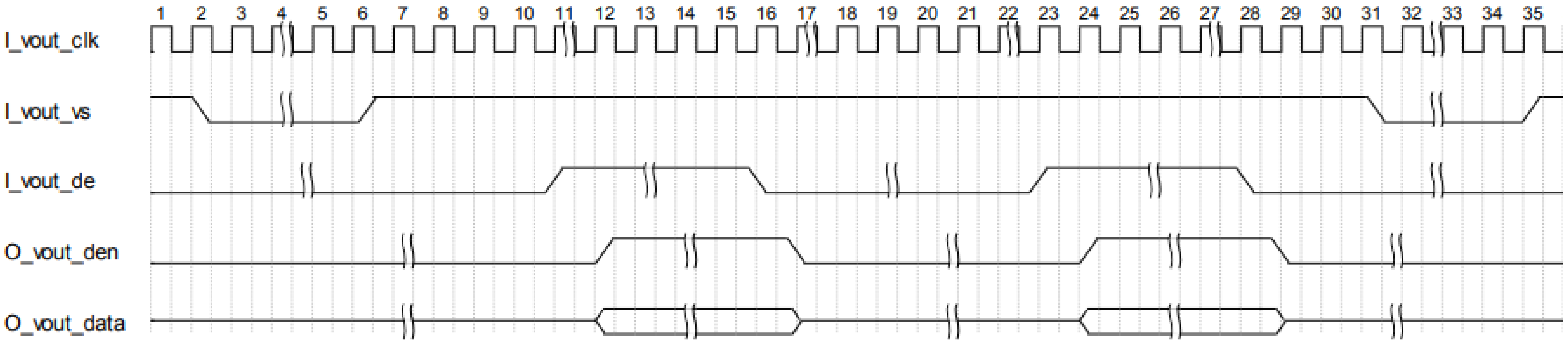


视频帧缓存控制器
I_vin_clk
I_vin_vs
I_vin_de
I_vin_data
输入行缓存控制电路
基地址切换控制电路
仲裁器控制电路
输出行缓存控制电路
I_vout_clk
I_vout_vs
I_vout_de
O_vout_den
O_vout_data
存储器接口控制器IP
外部存储器

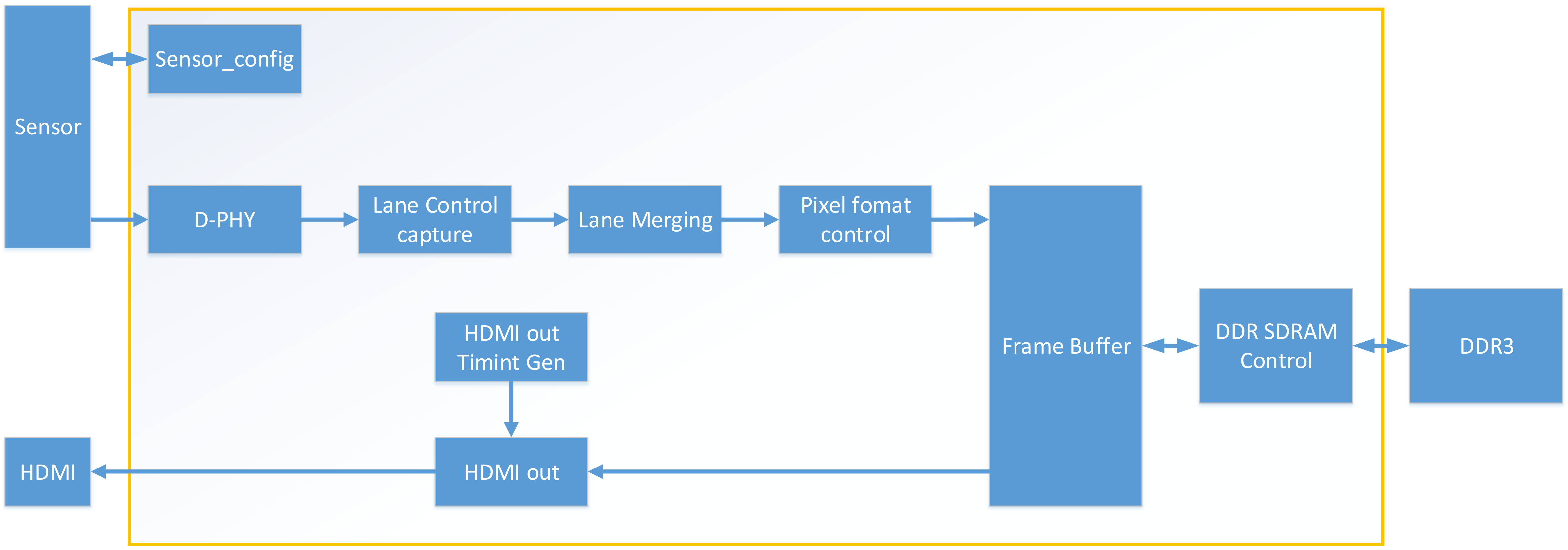
帧缓冲视频流时序



I_vin0_de 在一行内必须连续，不支持一行内 DE 不连续。



系统设计功能模块框图



Sensor
Sensor_config
D-PHY
Lane Control capture
Lane Merging
Pixel fomat control
Frame Buffer
DDR SDRAM Control
DDR3
HDMI out Timint Gen
HDMI out
HDMI

GOWIN高云

智慧逻辑 定制未来

MYMINIEYE

Look to the future together

深圳市我是你的眼有限公司

专业的FPGA、无线通信方案商

MYMINIEYE