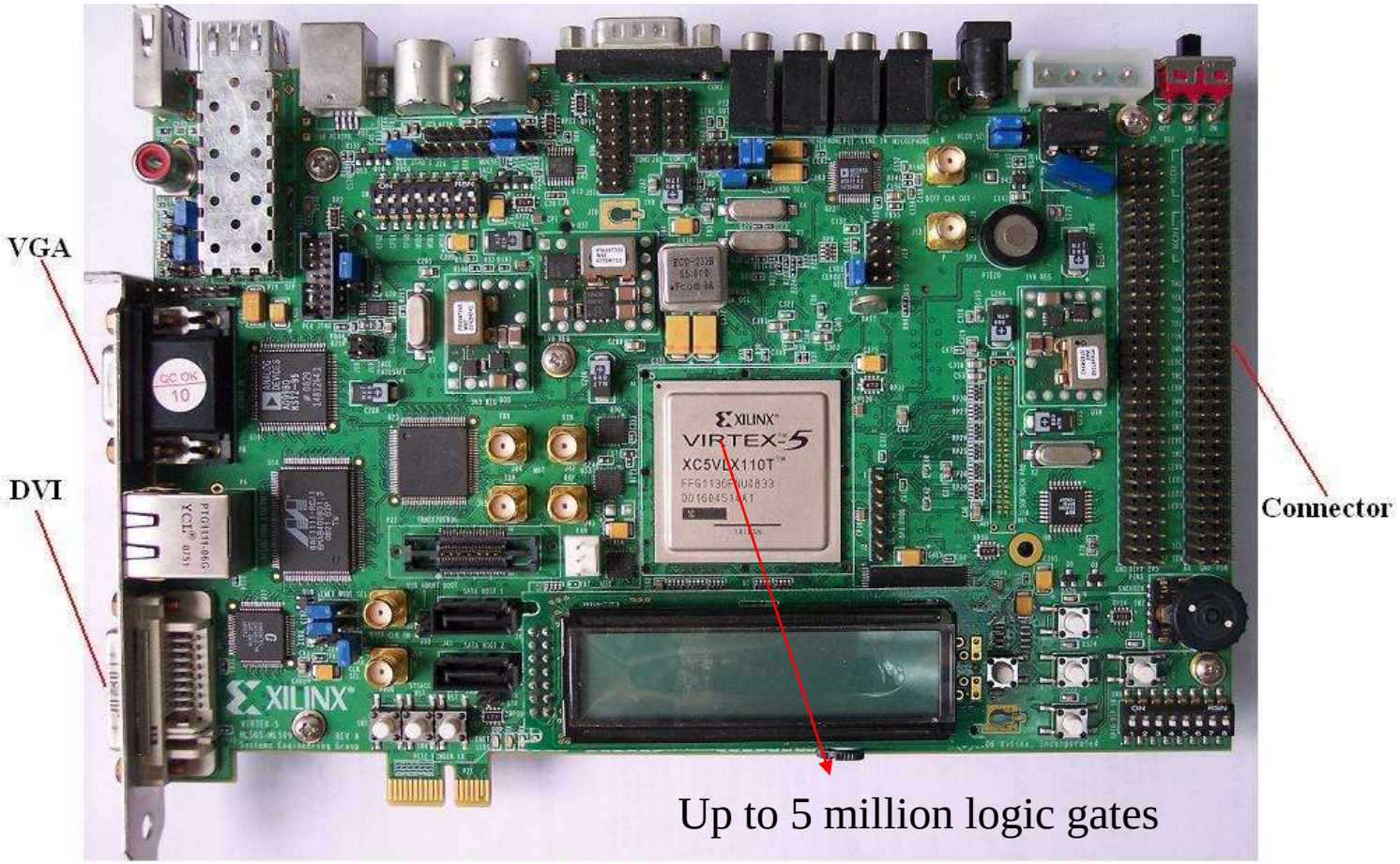
XUP V5-LX110T视频输入/输出方案交流

杨胜齐，吴磊，王杜瑶

交流内容

- 视频输入/输出方案—VGA DVI
- 视频输入/输出方案—Analog/Digital Video Daughter Cards DVI
- 视频输入/输出方案— PCIe DVI

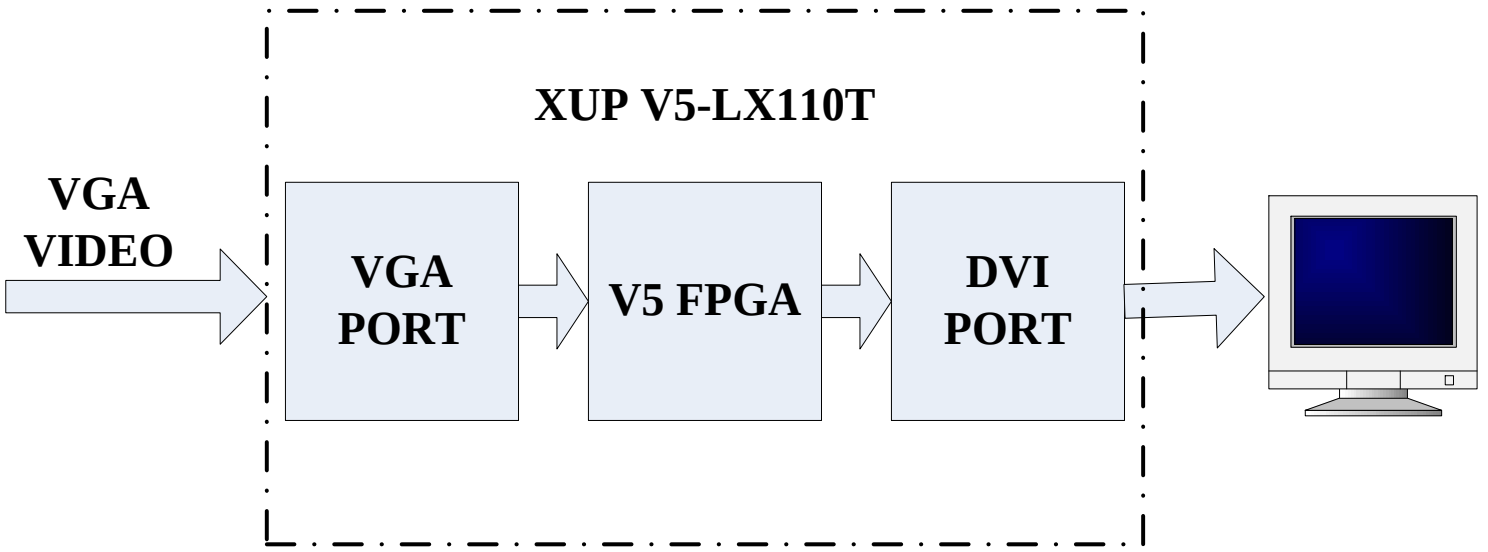
V5母板特点



XUP V5-LX110T FPGA平台

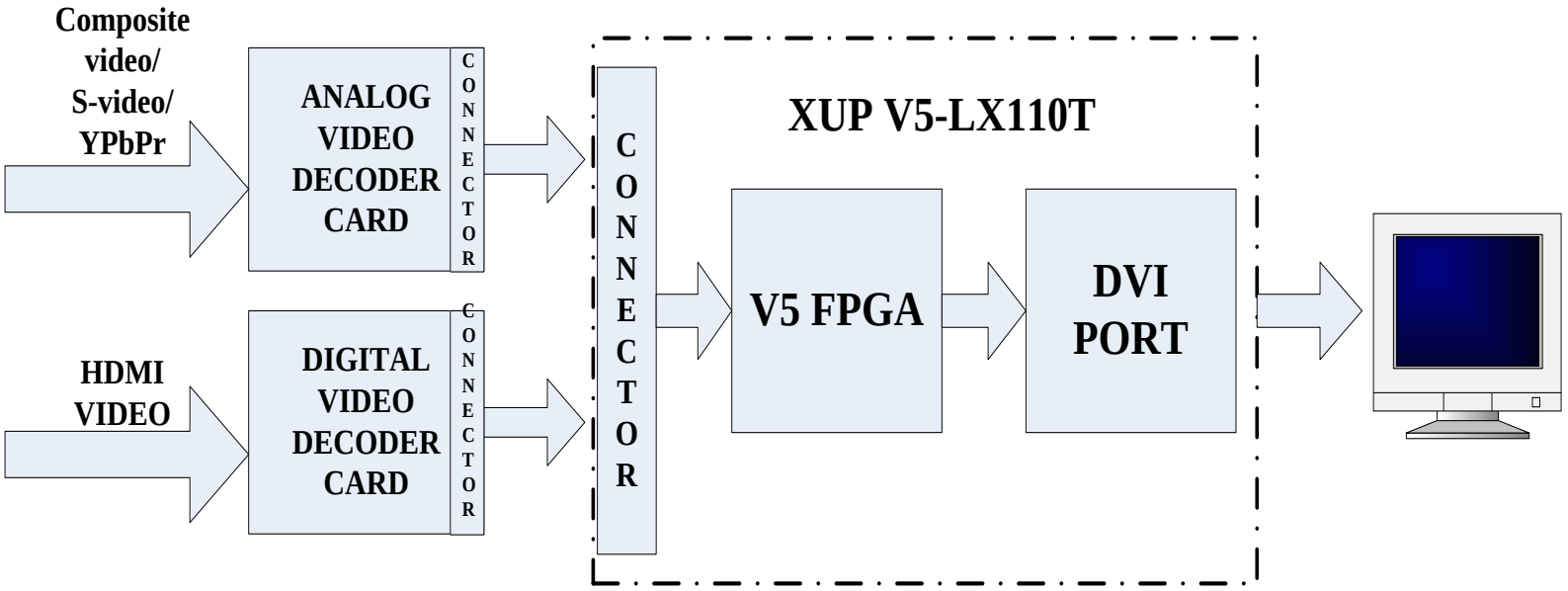
视频输入输出

- 板内方案



视频输入输出

- 板间方案


Composite video/ S-video/ YPbPr
ANALOG VIDEO DECODER CARD
CONNECTOR
HDMI VIDEO
DIGITAL VIDEO DECODER CARD
CONNECTOR
XUP V5-LX110T
CONNECTOR
V5 FPGA
DVI PORT

视频输入输出

- 板与PC方案

视频输入输出– 实验要求

- 硬件需求 V5 FPGA板与子板
  VGA视频输入源
  模拟/数字视频输入源
  视频输出显示器
- 软件需求：ISE 10.1与EDK

交流内容

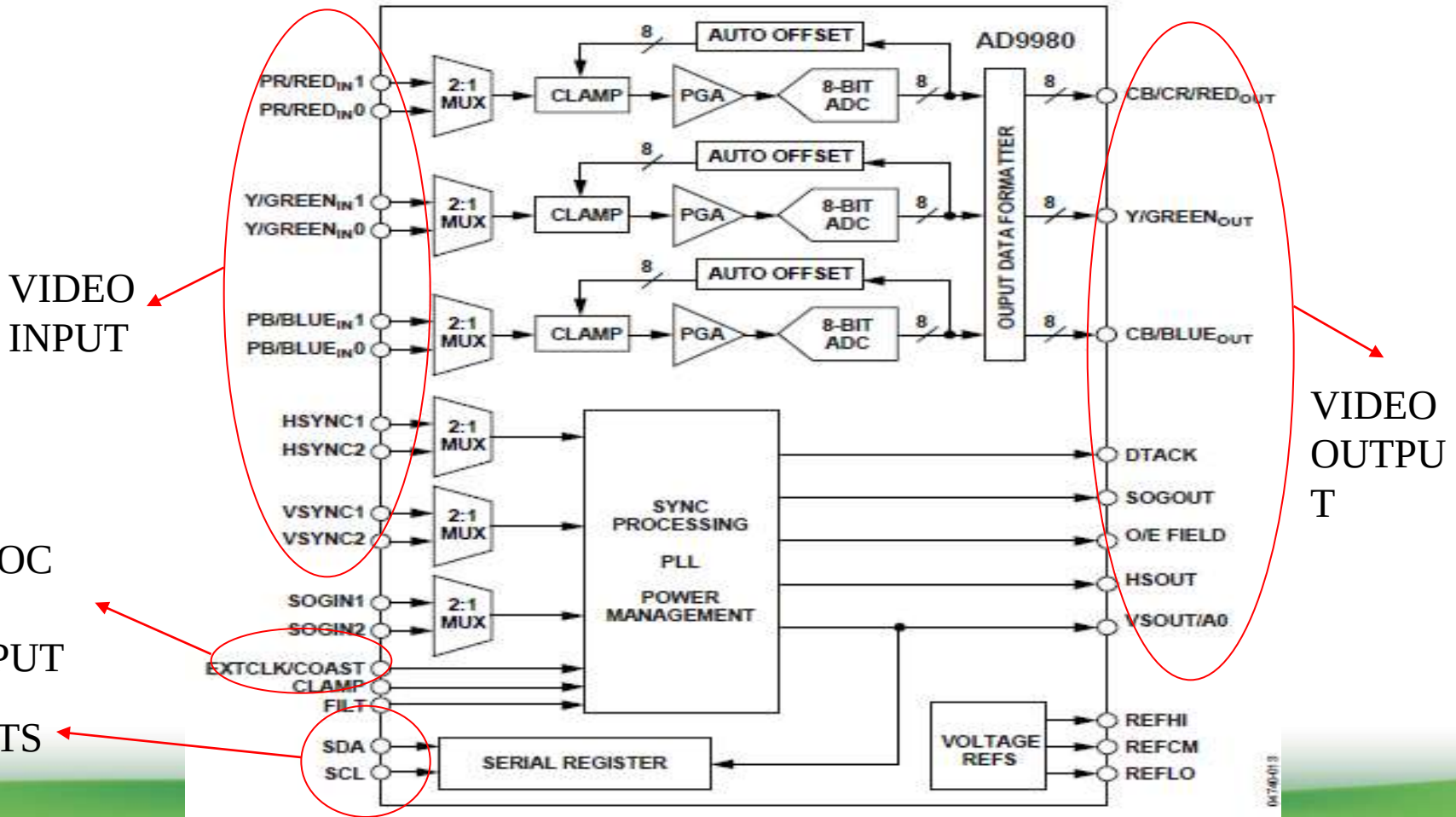
- 视频输入/输出方案—VGA DVI
- 视频输入/输出方案—Analog/Digital Video Daughter Cards DVI
- 视频输入/输出方案— PCIe DVI

VGA芯片的特点

- 视频控制芯片 VGA AD9980
  - 高性能8bit显示接口
  - 95MSPS最大转换率
  - 2:1输入多路复用
  - 4:4:4, 4:2:2和DDR输出格式模式
  - 奇偶场检测
  - 外部时钟输入
  - 再生输出HSYNC
  - IIC总线接口

VGA芯片的特点

- AD9980功能框图



VGA芯片的特点

- AD9980视频输入格式

Standard	Resolution	Refresh Rate (Hz)	Horizontal Frequency (kHz)	Pixel Rate (MHZ)	PLL Divider	VCORNGE	Current
VGA	640 × 480	60	31.500	25.175	800	01	100
		72	37.700	31.500	832	01	100
		75	37.500	31.500	840	01	100
		85	43.300	36.000	832	01	101
SVGA	800 × 600	56	35.100	36.000	1024	01	101
		60	37.900	40.000	1056	01	100
		72	48.100	50.000	1040	10	100
		75	46.900	49.500	1056	10	100
		85	53.700	56.250	1048	10	100
XGA	1024 × 768	60	48.400	65.000	1344	10	101
		70	56.500	75.000	1328	10	110
		75	60.000	78.750	1312	10	110
		80	64.000	85.500	1336	11	100
		85	68.300	94.500	1376	11	100

VGA芯片的特点

- **AD9980视频输出格式**

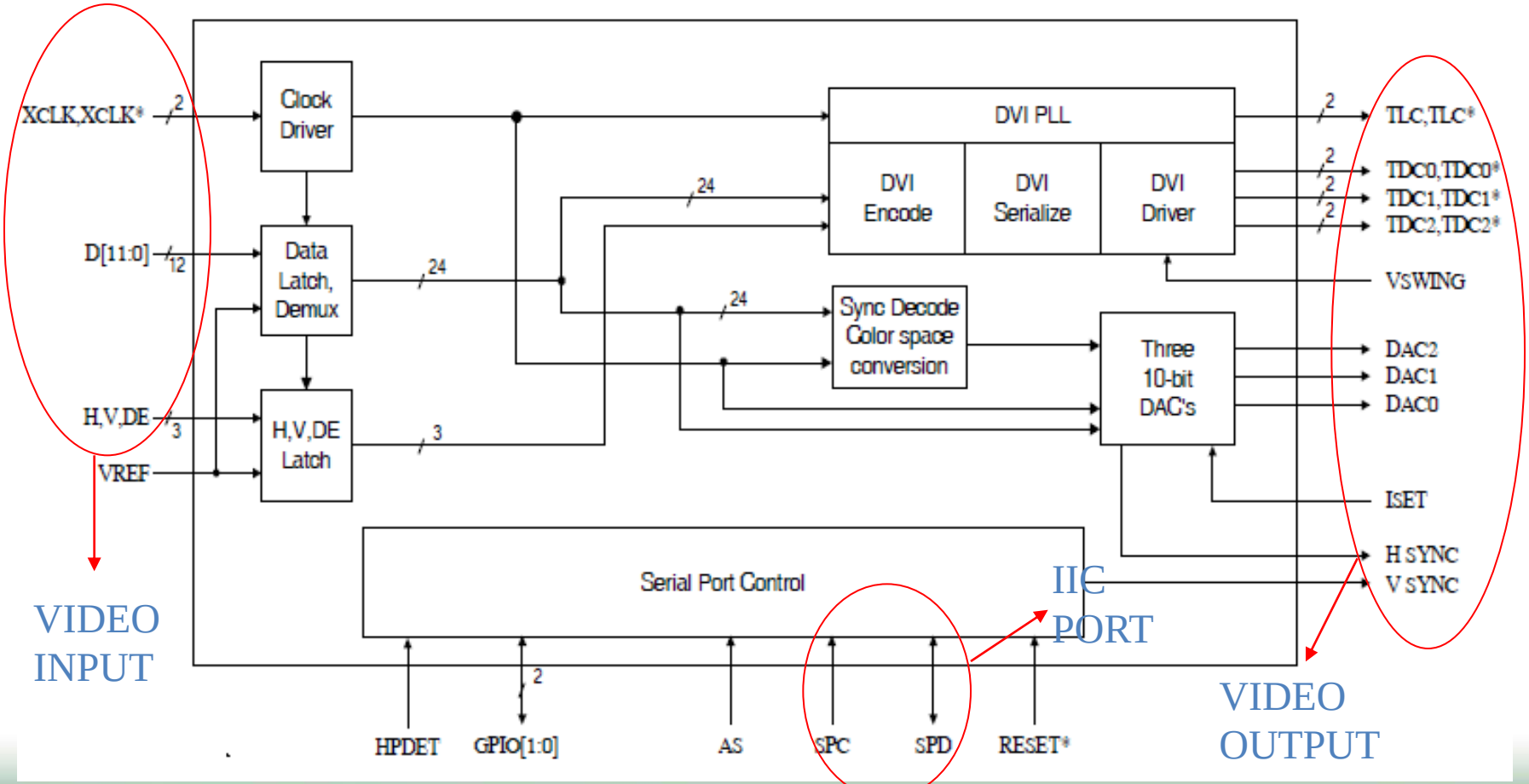
Port	Red								Green								Blue							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
4:4:4	Red/Cr								Green/Y								Blue/Cb							
4:2:2[1]	Cb, Cr								Y								DDR 4:2:2 ↑ Cb, Cr ↓ Y, Y							
4:4:4 DDR	DDR ↑[2] G [3:0]				DDR ↑ B [7:4]				DDR ↑ B [3:0]				N/A				DDR 4:2:2 ↑ Cb, Cr							
	DDR ↓ R [7:0]								DDR ↓ G [7:4]				N/A				DDR 4:2:2 ↓ Y, Y							

DVI芯片的特点

- DVI 发送芯片CH7301C
  - DVI发送速率高达165Mps
  - DVI热插拔检测
  - 支持图形分辨率最高达1600*1200
  - 提供RGB输出
  - DAC连接检测
  - 三路10bit视频DAC输出
  - IIC总线接口

DVI芯片的特点

- CH7301C功能框图



DVI芯片的特点

- **CH7301C视频输入格式**

0	12-bit multiplexed RGB input (24-bit color), (multiplex scheme 1)
1	12-bit multiplexed RGB2 input (24-bit color), (multiplex scheme 2)
2	8-bit multiplexed RGB input (16-bit color, 565)
3	8-bit multiplexed RGB input (15-bit color, 555)
4	8-bit multiplexed YCrCb input (24-bit color), (Y, Cr and Cb are multiplexed)

DVI芯片的特点

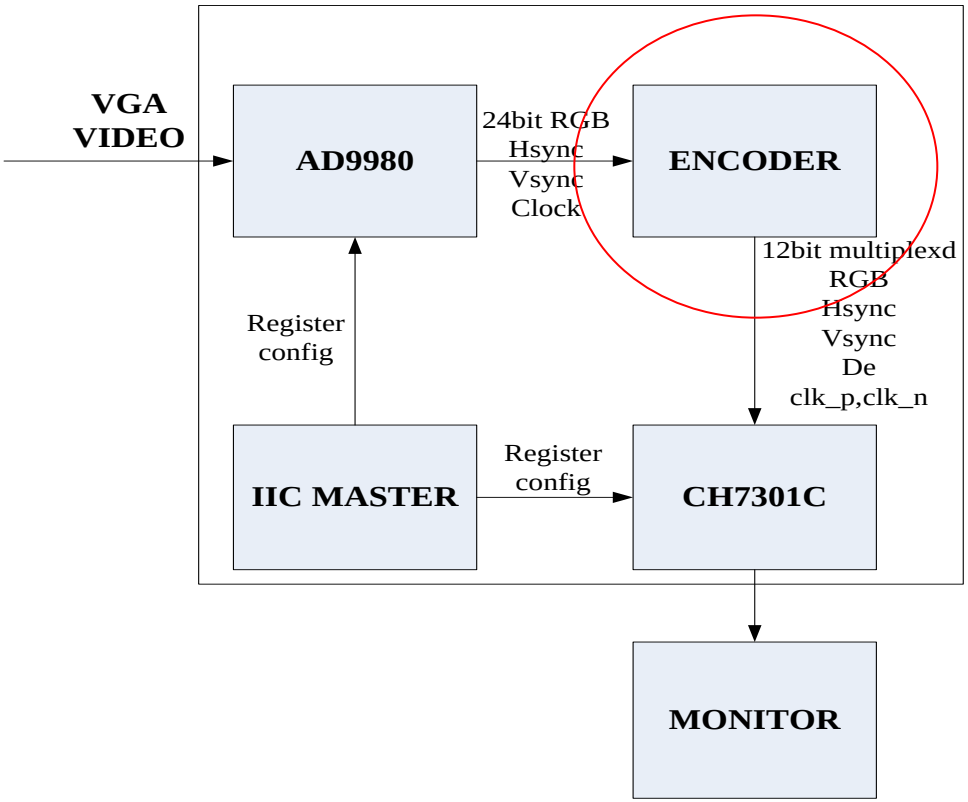
- CH7301C视频输出格式

Table 2. DVI Outputs

Graphics Resolution	Active Aspect Ratio	Pixel Aspect Ratio	Refresh Rate (Hz)	XCLK Frequency (MHz)	DVI Frequency (Mbits)
720x400	4:3	1.35:1.00	<85	<35.5	<355
640x400	8:5	1:1	<85	<31.5	<315
640x480	4:3	1:1	<85	<36	<360
720x480	4:3	9:8	59.94	27	270
720x576	4:3	15:12	50	27	270
800x600	4:3	1:1	<85	<57	<570
1024x768	4:3	1:1	<85	<95	<950
1280x720	16:9	1:1	<85	<110	<1100
1280x768	15:9	1:1	<85	<119	<1190
1280x1024	4:3	1:1	<85	<158	<1580
1366x768	16:9	1:1	<85	<140	<1400
1360x1024	4:3	1:1	<75	<145	<1450
1400x1050	4:3	1:1	<75	<156	<1560
1600x1200	4:3	1:1	<60	<165	<1650
1920x1080[1]	16:9	1:1	<60	<165	<1650

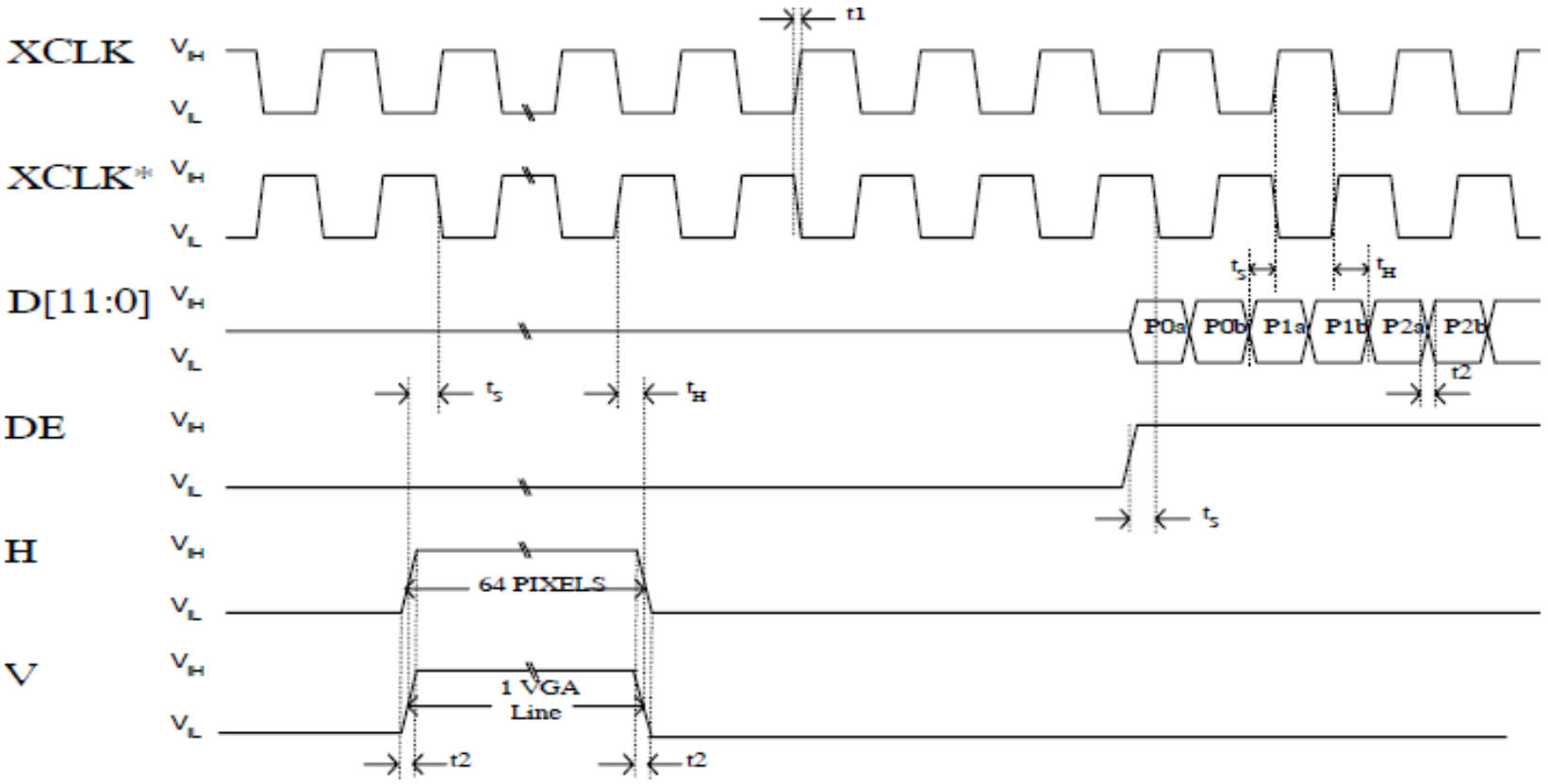
VGA-DVI输入输出设计

- IP设计流程图



VGA-DVI输入输出设计

- 编码模块



VGA-DVI输入输出设计

- 差分时钟的产生：Xilinx IP核DCM

```
DCM_BASE (
    .CLK0(),         // 0 degree DCM CLK output
    .CLK180(),       // 180 degree DCM CLK output
    .CLK270(),       // 270 degree DCM CLK output
    .CLK2X(),        // 2X DCM CLK output
    .CLK2X180(), // 2X, 180 degree DCM CLK out
    .CLK90(),        // 90 degree DCM CLK output
    .CLKDV(),        // Divided DCM CLK out (CLKDV_DIVIDE)
    .CLKFX(),        // DCM CLK synthesis out (M/D)
    .CLKFX180(), // 180 degree CLK synthesis out
    .LOCKED(),       // DCM LOCK status output
    .CLKFB(),        // DCM clock feedback
    .CLKIN(),        // Clock input (from IBUFG, BUFG or DCM)
    .RST()           // DCM asynchronous reset input
  );
```

VGA-DVI输入输出设计

- **12bit复用数据D[11:0]和DE信号产生**

```
assign de= !(h | v);
assign r = (de==1'b1 )? R[7:0]:8'b00000000;
assign g = (de==1'b1 )? G[7:0]:8'b00000000;
assign b = (de==1'b1 )? B[7:0]:8'b00000000;

 assign d[11:0]= (clk_p==1'b1 )? {r[7:0], g[7:4]}:{g[3:0], b[7:0]};
```

VGA-DVI输入输出设计

- Demo

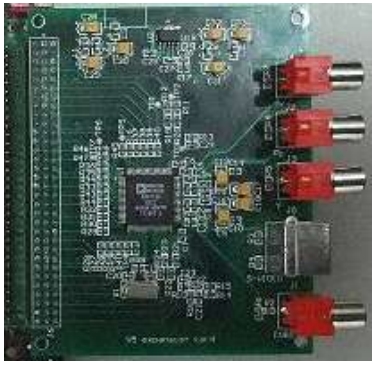
放一张demo的图片

交流内容

- 视频输入/输出方案—VGA DVI
- 视频输入/输出方案—Analog/Digital Video Daughter Cards DVI
- 视频输入/输出方案— PCIe DVI

模拟/数字视频输入子板

- 视频解码输入子板



  - 支持模拟视频输入的子板（已完成）

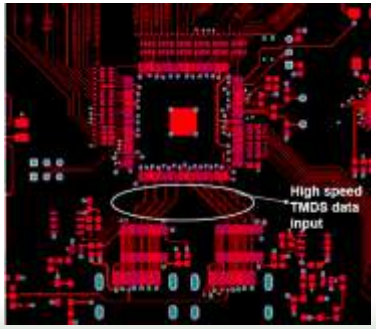
    Composite video

    S-video

    YPrPb

  - 支持数字视频输入的子板（正在调试）

    HDMI



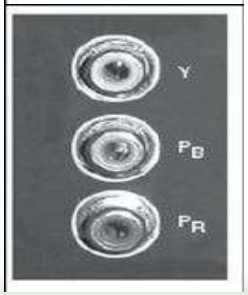
模拟/数字视频输入子板

- 输入模式

  - CVBS：Composite video broadcast signal，复合视频。基带信号，将亮度、色度、同步和色彩脉冲信息整合到一根电缆内

  - S-video：分别传送亮度和色度信号。将亮度和色度信号分离传送能大幅改善图像质量

  - YPrPb：分量视频信号。每个亮度和色度通道都是单独提取、输出，每路都带有自己的时序

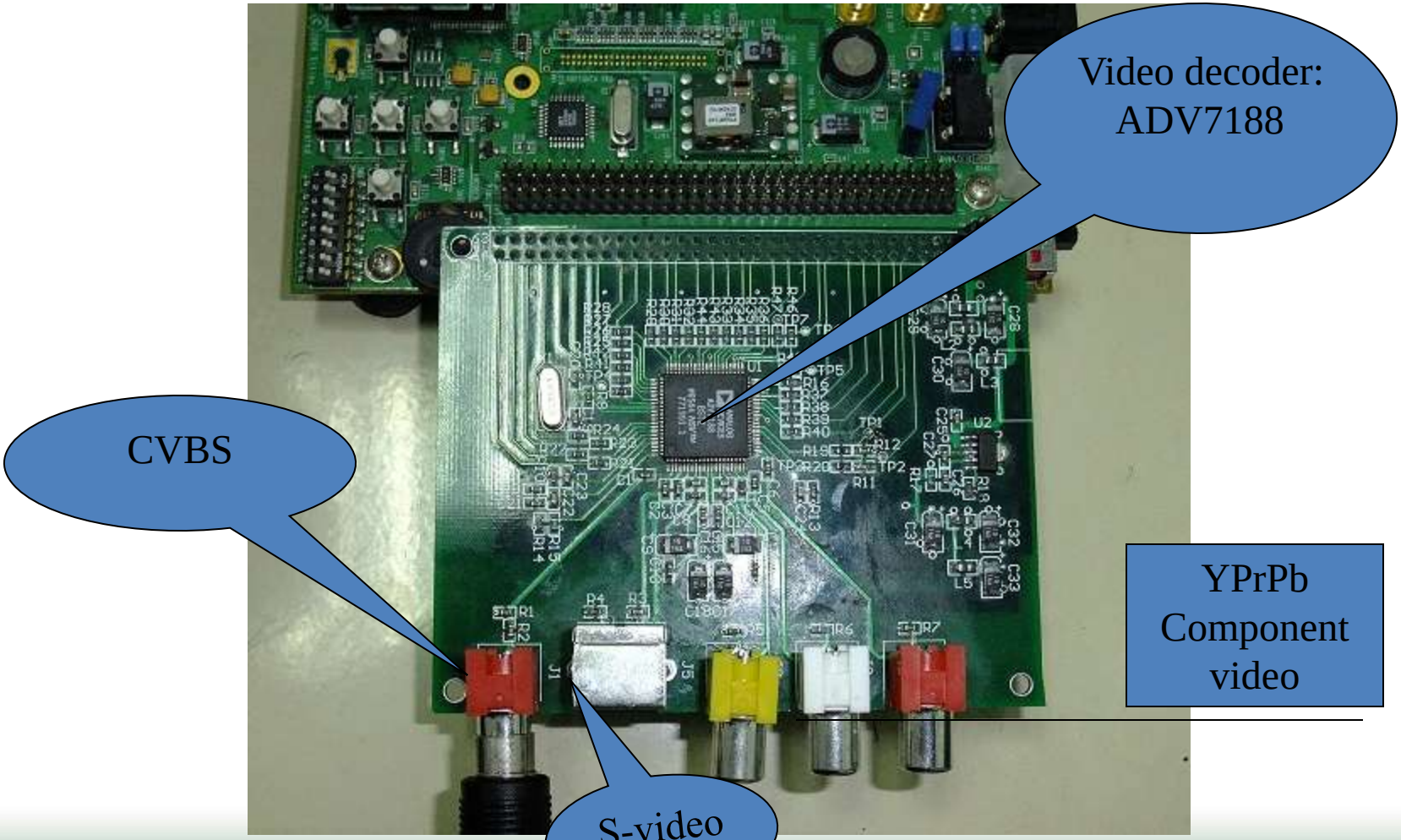


模拟/数字视频输入子板

- **解码芯片ADV7188**
  - 自动检测NTSC/PAL/SECAM制式电视信号
  - 解码CVBS、S-video和YPrPb等视频信号

  - 输出为ITU-R BT.656标准的8/10/ 16/20-bit YCrCb 4:2:2数据和行、场以及奇偶场同步信号。

模拟/数字视频输入子板



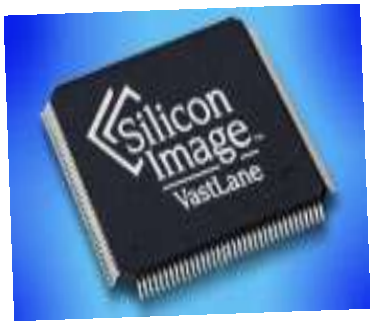
模拟/数字视频输入子板

- 数字视频解码子板的设计

HDMI: 能高品质地传输未经压缩的高清视频和多声道音频数据，最高数据传输速度为5Gbps。

模拟/数字视频输入子板

- SIL9135解码芯片
  1080p@60Hz
  720p/1080i@120Hz
  36位彩色深度。



模拟视频Preliminaries

- 视频控制信号

HSYNC是水平同步信号。它界定了视频帧每一行中（从左到右）有效视频的起始位置。水平消隐为电子枪从屏幕右侧回扫至下一行左侧的时间间隔。

VSYNC是垂直同步信号。它定义了一个新的视频图像的起始位置（从上到下）。垂直消隐为电子枪从屏幕图像的右下角返回左上角所需的时间间隔。

FIELD用于在隔行视频信号中区分出目前所显示的场。该信号并不适用于逐行扫描视频系统。

模拟视频Preliminaries

- 场消隐定义

		625	525
V-digital field blanking			
Field 1	Start (V = 1)	Line 624	Line 1
	Finish (V = 0)	Line 23	Line 20
Field 2	Start (V = 1)	Line 311	Line 264
	Finish (V = 0)	Line 336	Line 283
F-digital field identification			
Field 1	F = 0	Line 1	Line 4
Field 2	F = 1	Line 313	Line 266

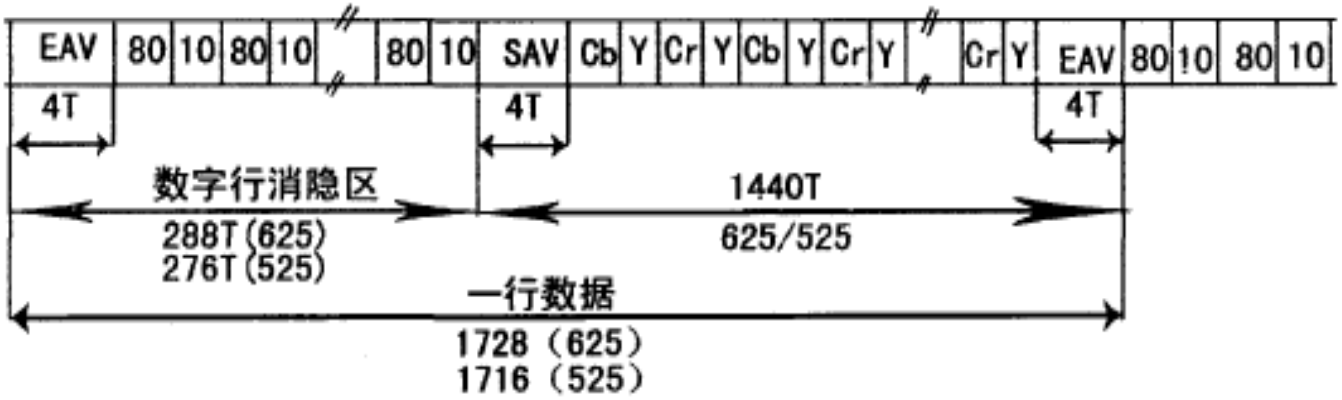
模拟视频Preliminaries

- **ITU-R BT.656格式**

ITU656的数据信息是以8比特字的二进制信息的形式，通过串行的方式来传输亮度、信号(Y)以及色度信号(Cbcr)，同时传输 视频时序参考信号

模拟视频Preliminaries

- **ITU-R BT656 数据格式**



T: 时钟周期37ns

SAV: start of active video

EAV: end of active video

模拟视频Preliminaries

- **SAV、EAV定义**

由四个字节组成：FF、00、00、XY。其中前三个是固定字节，最后的XY包含了视频控制信号。

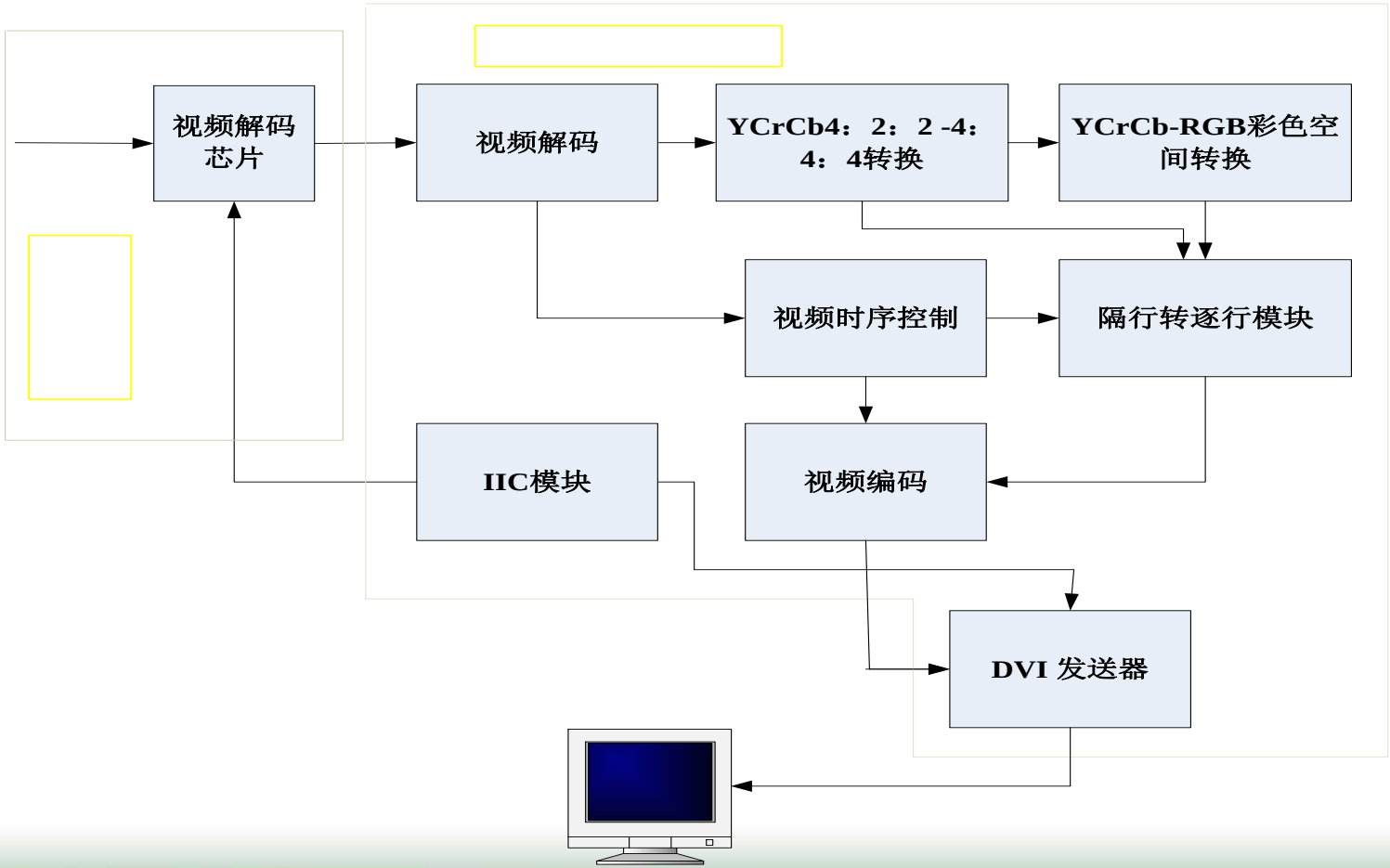
Data bit number	First word (FF)	Second word (00)	Third word (00)	Fourth word (XY)
9 (MSB)	1	0	0	1
8	1	0	0	F
7	1	0	0	V
6	1	0	0	H
5	1	0	0	P_3
4	1	0	0	P_2
3	1	0	0	P_1
2	1	0	0	P_0
1 (Note 2)	1	0	0	0
0	1	0	0	0

- 模拟视频子板工作原理

IIC模块通过IIC总线配置ADV7188芯片，从而将摄像头或者DVD播放机输入的电视制式27MHz复合视频转化为数字8bit ITU-R BT.656视频标准格式。这种数据流包含了时序信号和YCrCb格式信号。

模拟视频输入输出

- 模拟视频输入的子板系统IP设计流程图

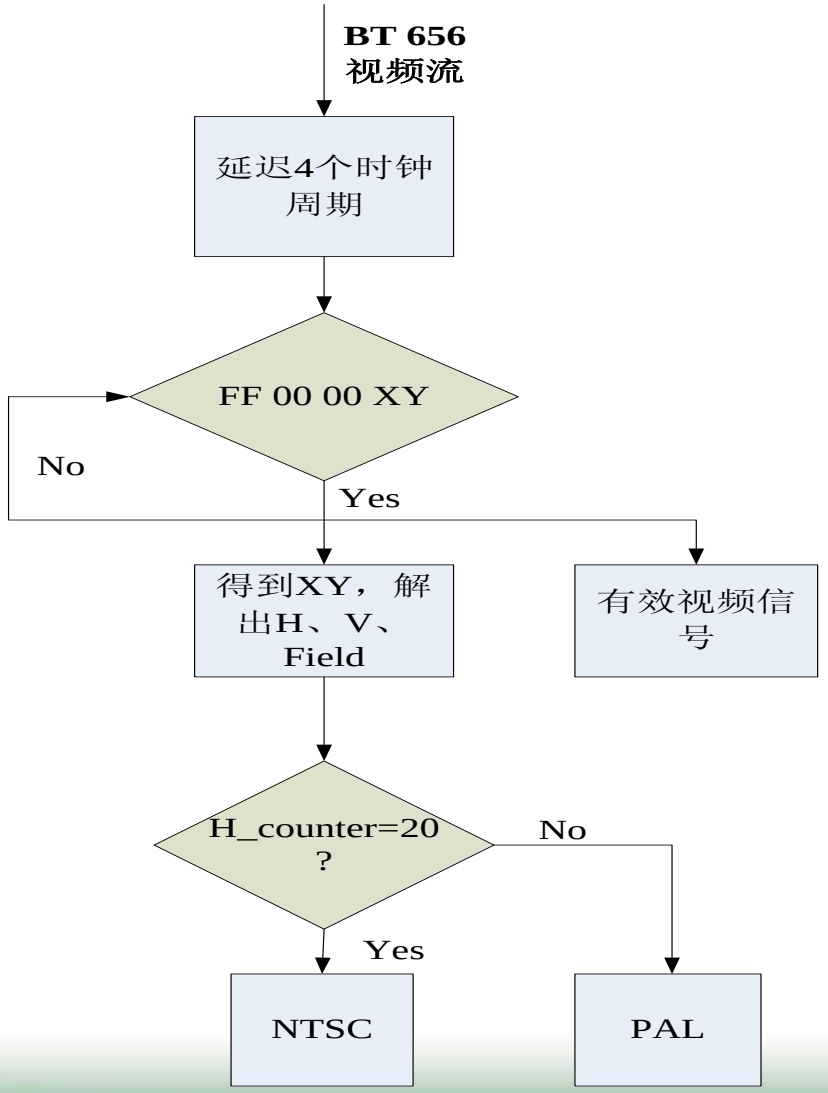


模拟视频输入输出

- 视频解码模块
  - ADV7188输出8位ITU-R BT.656的YCrCb型4:2:2视频数据；分辨率为640x480；输出像素时钟为27MHz。
  - 该模块功能就是提取视频流中的视频控制信号Hsync、Vsync以及Field信号。

模拟视频输入输出

- 视频解码模块框图



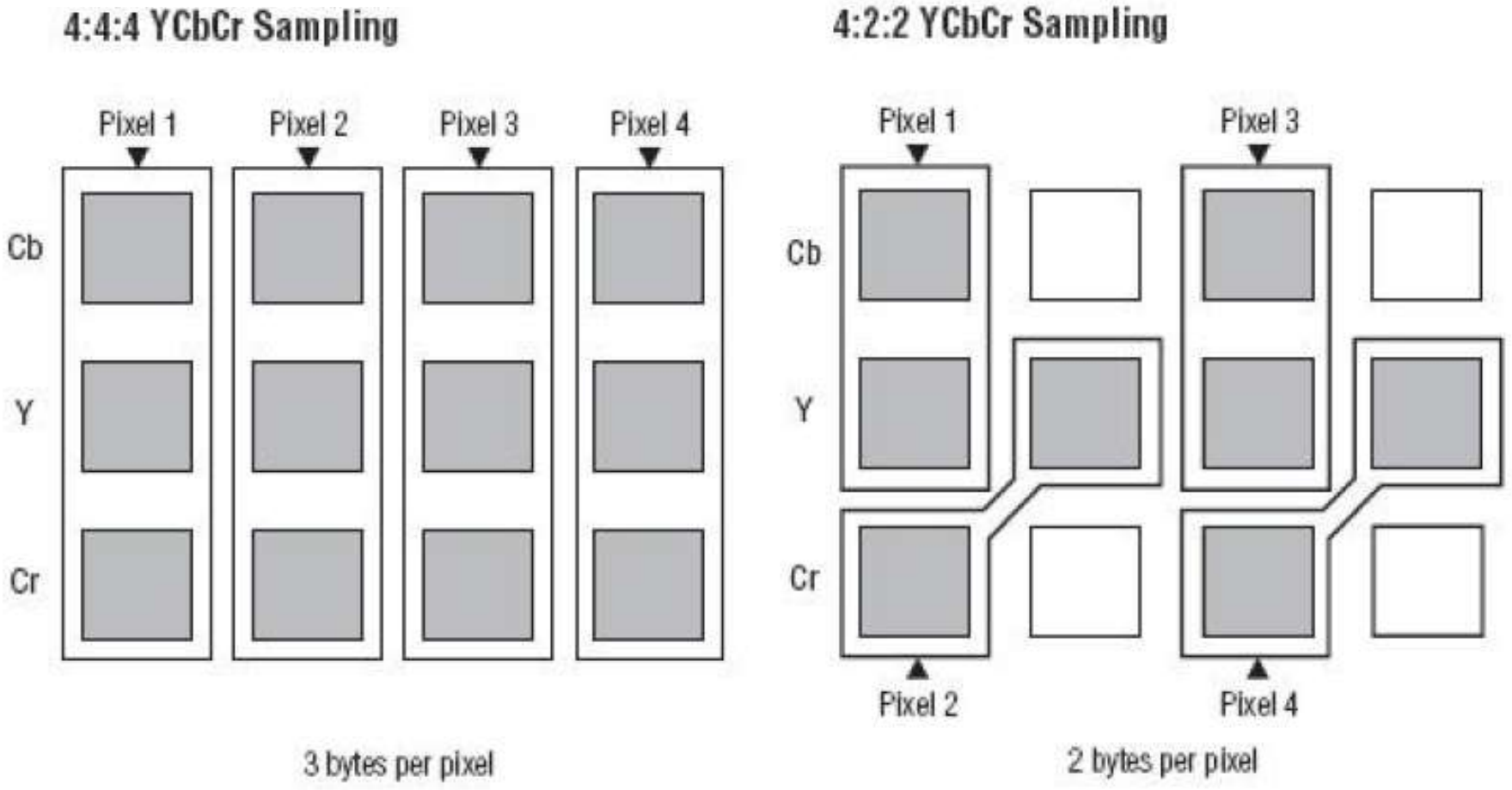
模拟视频输入输出

- **4:2:2 to 4:4:4数据转换模块**

  - ITU-R BT.656 采用的是4:2:2的标准，也就是每采4个亮度信号，就各采样2个色差信号

  - 因此在进行色彩空间变换之前，需要把丢失的2个色差信号补回来。弥补的方法是通过复制相邻像素点的色差信息来实现。

模拟视频输入输出


4:4:4 YCbCr Sampling
Pixel 1
Pixel 2
Pixel 3
Pixel 4
Cb
Y
Cr
3 bytes per pixel
4:2:2 YCbCr Sampling
Pixel 1
Pixel 3
Cb
Y
Cr
Pixel 2
Pixel 4
2 bytes per pixel

模拟视频输入输出

- 解码出来的数据格式为Cb Y Cr Y……，所以可以设计一个2bit计数器，其中有四个状态00，01，10，11。当为01和11时，对亮度信号Y进行采样；当为00时对Cb信号进行采样；而当为10时对Cr信号进行采样。

```
assign ena_luma_reg   =  ((state_cnt[0] & ~state_cnt[1]) | (state_cnt[1]
    &                    state_cnt[0]));//counts 1, 3
assign ena_chroma_blue_reg =  (~state_cnt[0] & ~state_cnt[1]);
    //count 0
assign ena_chroma_red_reg  =  (~state_cnt[0] &  state_cnt[1]);
    //count 2
```

模拟视频输入输出

- **YCrCb2RGB转换模块**

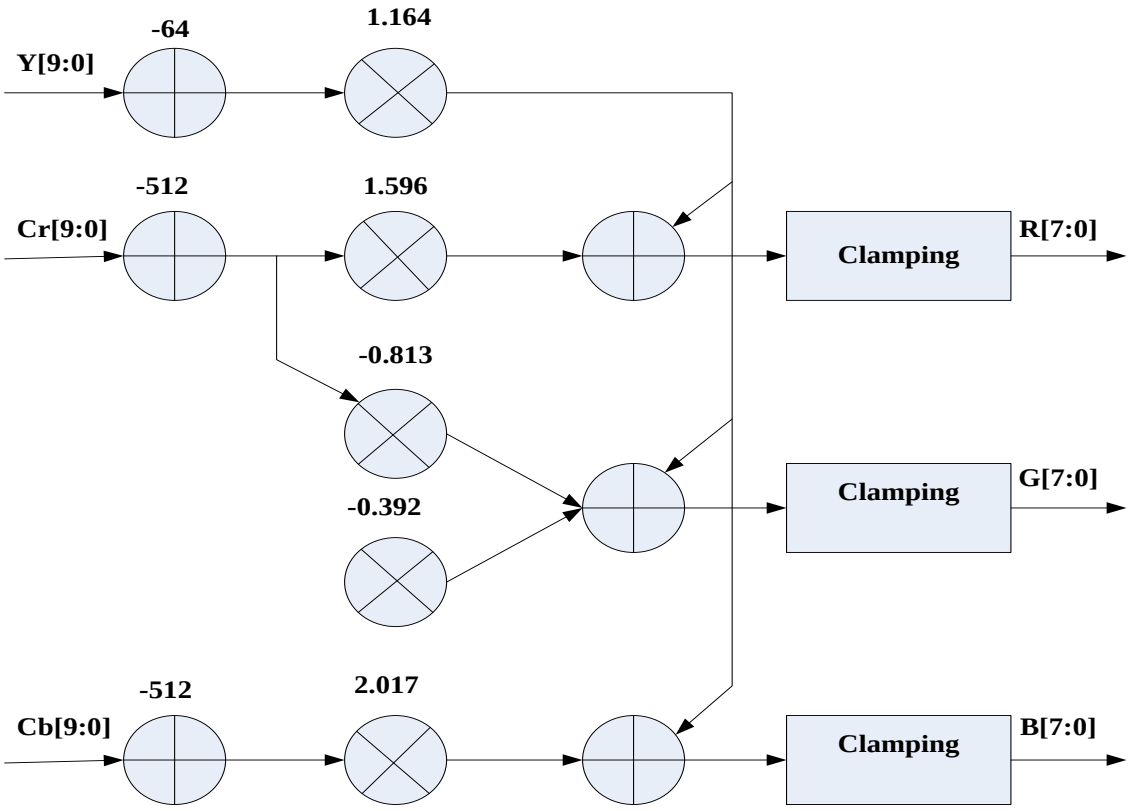
$$R=1.164(Y-64)+1.596(Cr-512)$$

$$G=1.164(Y-64)-0.813(Cr-512)-0.392(Cb-512)$$

$$B=1.164(Y-64)+2.017(Cb-512)$$

模拟视频输入输出

- YCrCb2RGB IP 转换算法原理图



模拟视频输入输出

- **YCbCr 2 RGB 部分代码**

小数部分转换

```
const1 = 10'b 0100101010; //1.164 = 01.00101010
const2 = 10'b 0110011000; //1.596 = 01.10011000
const3 = 10'b 0011010000; //0.813 = 00.11010000
const4 = 10'b 0001100100; //0.392 = 00.01100100
const5 = 10'b 1000000100; //2.017 = 10.00000100
```

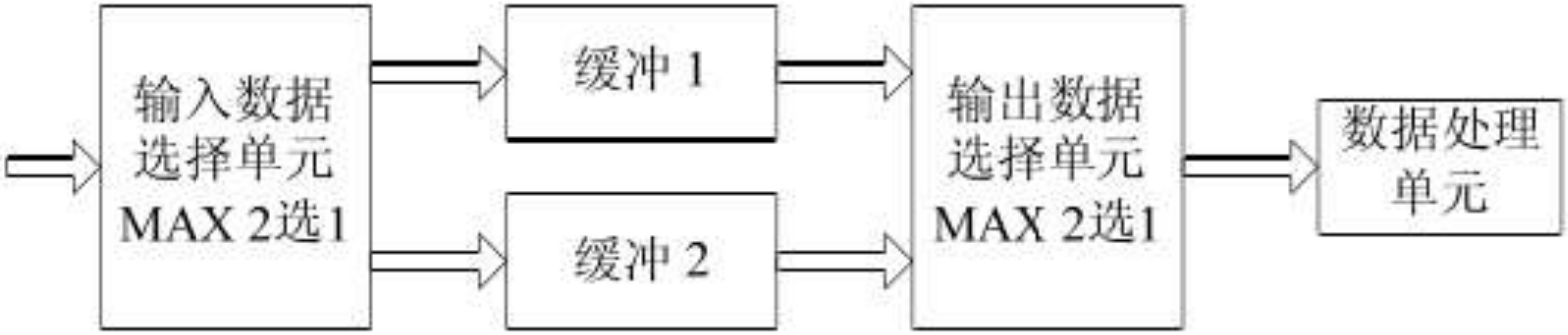
最后结果采用截数得到

```
assign R = (R_int[20]) ? 0 : (R_int[19:18] == 2'b0) ? R_int[17:10] : 8'b11111111;
assign G = (G_int[20]) ? 0 : (G_int[19:18] == 2'b0) ? G_int[17:10] : 8'b11111111;
assign B = (B_int[20]) ? 0 : (B_int[19:18] == 2'b0) ? B_int[17:10] : 8'b11111111;
```

模拟视频输入输出

- 隔行转逐行模块

由于解码得到的视频信号为隔行信号，而DVI发送器只接受逐行信号。本方案采用行复制的方法，达到隔行转逐行的目的。这里通过乒乓操作就可以实现。



模拟视频输入输出

- **Line buffer设计**

Line buffer 当作数据缓冲模块，用来缓存一行视频信号的数据。Line buffer由三个Block RAM构成，这三个块RAM分别存取R,G,B三路信号。

模拟视频输入输出

- **Block RAM: 2K*8缓存空间例化**

```
RAMB16_S9_S9_inst (
   .DOA(DOA),      // Port A 8-bit Data Output
   .DOB(DOB),      // Port B 8-bit Data Output
   .DOPA(DOPA),    // Port A 1-bit Parity Output
   .DOPB(DOPB),    // Port B 1-bit Parity Output
   .ADDRA(ADDRA),  // Port A 11-bit Address Input
   .ADDRB(ADDRB),  // Port B 11-bit Address Input
   .CLKA(CLKA),    // Port A Clock
   .CLKB(CLKB),    // Port B Clock
   .DIA(DIA),      // Port A 8-bit Data Input
   .DIB(DIB),      // Port B 8-bit Data Input
   .DIPA(DIPA),    // Port A 1-bit parity Input
   .DIPB(DIPB),    // Port-B 1-bit parity Input
   .ENA(ENA),      // Port A RAM Enable Input
   .ENB(ENB),      // Port B RAM Enable Input
   .SSRA(SSRA),    // Port A Synchronous Set/Reset Input
   .SSRB(SSRB),    // Port B Synchronous Set/Reset Input
   .WEA(WEA),      // Port A Write Enable Input
   .WEB(WEB)       // Port B Write Enable Input
);
```

模拟视频输入输出

- 例化R信号

```
RAMB16_S9_S9 RED_DATA_RAM(
        .DOA(),
        .DOPA(),
        .ADDRA(write_address[10:0]),
        .CLKA(write_clk),
        .DIA(write_red_data[7:0]),
        .DIPA(1'b0),
        .ENA(write_enable),
        .WEA(1'b1),
        .SSRA(1'b0),

        .DOB(read_red_data[7:0]),
        .DOPB(),
        .ADDRB(read_address[10:0]),
        .CLKB(read_clk),
        .DIB(8'h00),
        .DIPB(1'b0),
        .ENB(read_enable),
        .SSRB(1'b0),
        .WEB(1'b0)
    );
```

模拟视频输入输出

- 视频时序控制模块

根据输入视频制式，用计数方式产生Hsync、 Vsync信号，并延迟相应的时间周期保持与视频数据同步。

模拟视频输入输出

- **NTSC数据格式定义**

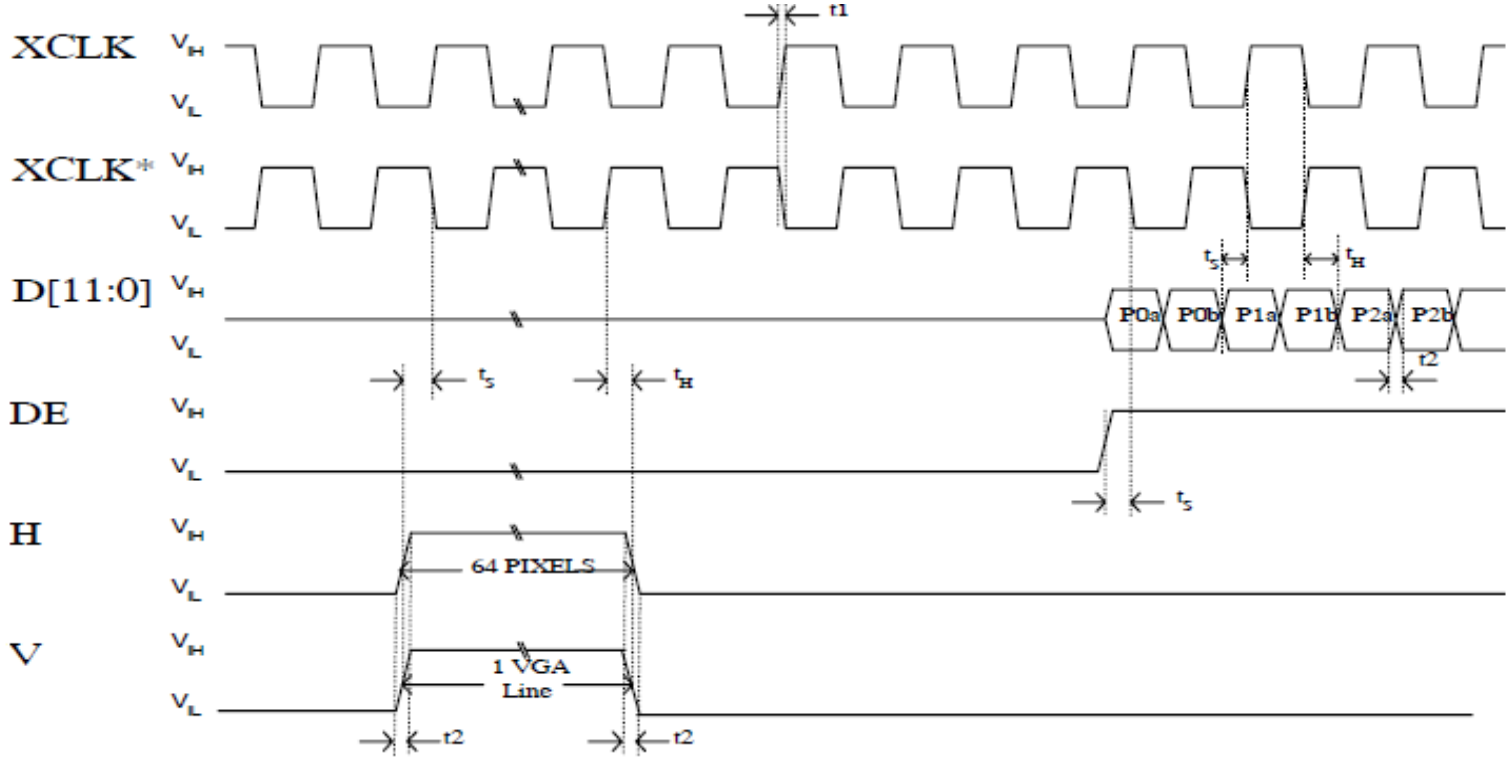
720 X 480 @ 60Hz with a 27MHz pixel clock for NTSC video capture

```
`define H_ACTIVE          720     // pixels
`define H_FRONT_PORCH     7       // pixels
`define H_SYNCH           62      // pixels
`define H_BACK_PORCH      69      // pixels
`define H_TOTAL           858 // pixels

`define V_ACTIVE          487     // lines
`define V_FRONT_PORCH     4       // lines
`define V_SYNCH           4       // lines
`define V_BACK_PORCH      30      // lines
`define V_TOTAL           525     // lines
```

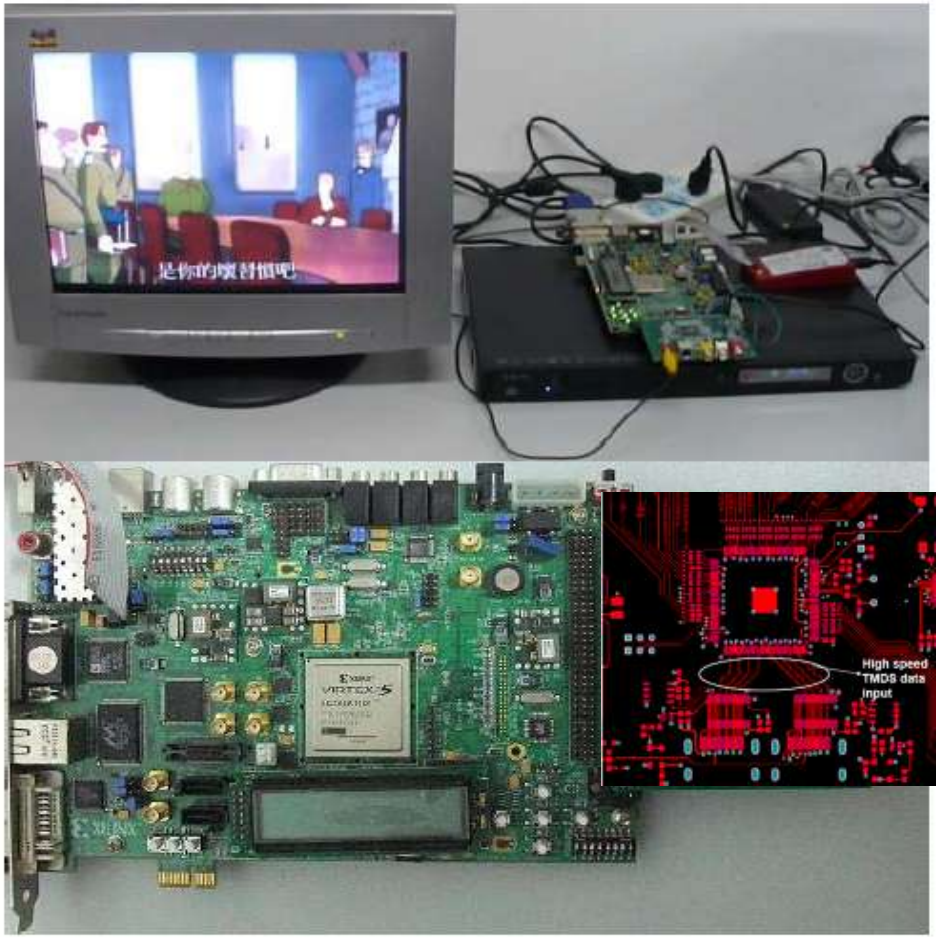
模拟视频输入输出

- 编码模块：与板内方案VGA视频输入，DVI视频输出的设计一样



模拟视频输入输出

敬请关注： HDMI Daughter Card



交流内容

- 视频输入/输出方案—VGA DVI
- 视频输入/输出方案—Analog/Digital Video Daughter Cards DVI
- 视频输入/输出方案— PCIe DVI

基于V5 FPGA
医学图像三维重建

基于王杜瑶的本科毕业设计

研究背景、意义

国内外研究现状

主要研究内容

下一步工作

总结

研究背景和意义

医学影像发展历程

1895:X射线

1972:CT

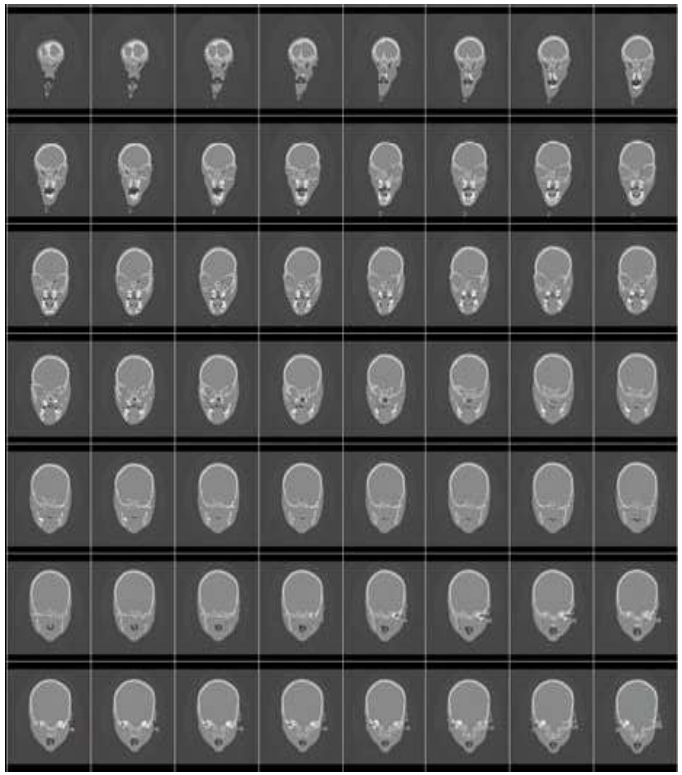
1985:MRI

1990’s:PET

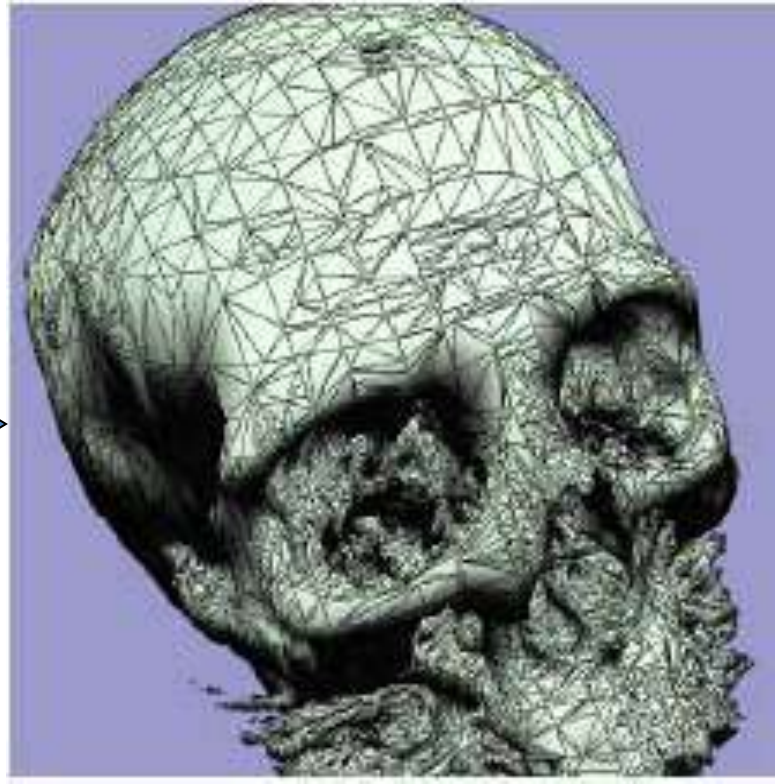
研究背景和意义

三维重建技术

二维图像



三维重建



三维实体

研究背景、意义

国内外研究现状

主要研究内容

下一步工作

总结

国内外研究现状

医学图像三维重建发展阶段

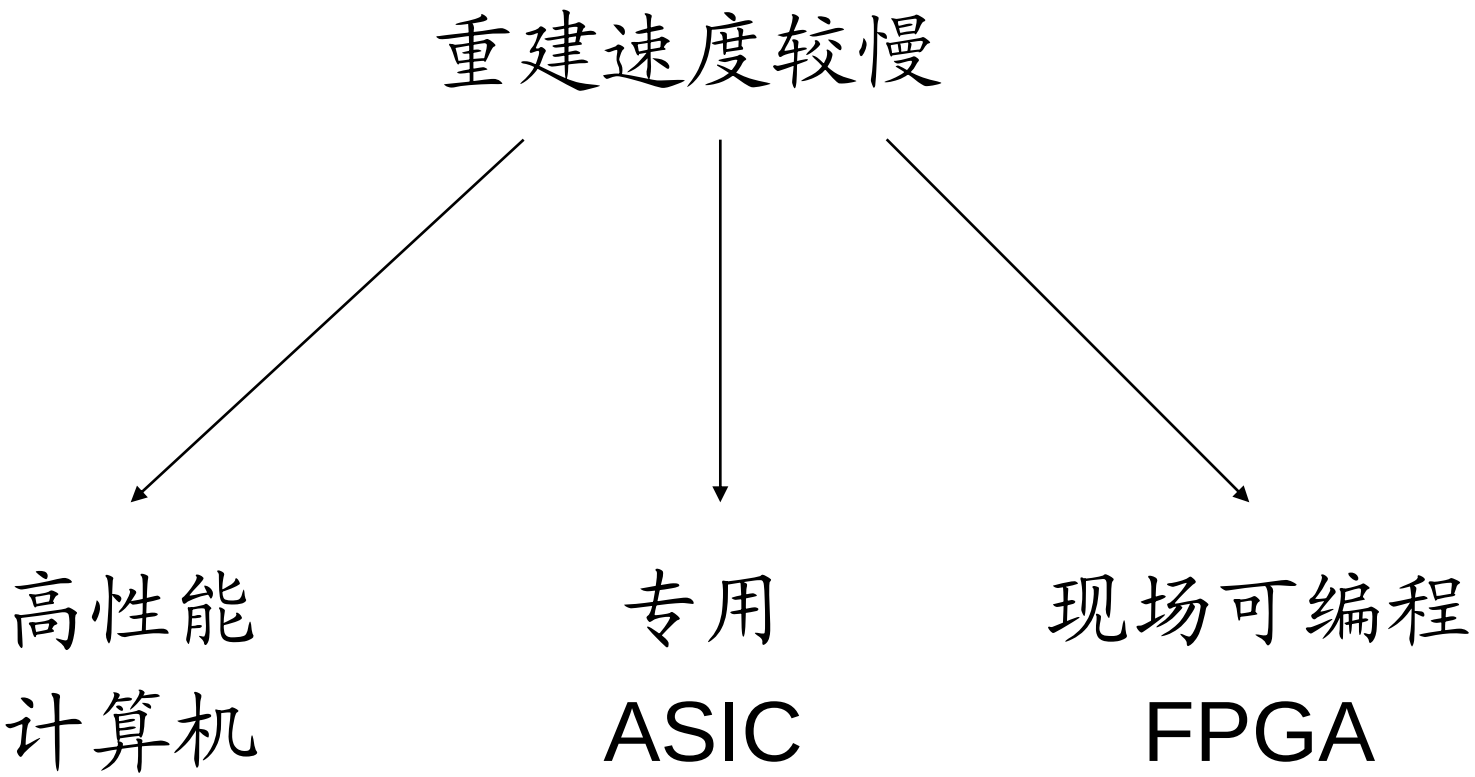
早期探索阶段

基础算法研究

实用系统研究

国内外研究现状

医学图像三维重建发展瓶颈



研究背景、意义

国内外研究现状

主要研究内容

下一步工作

总结

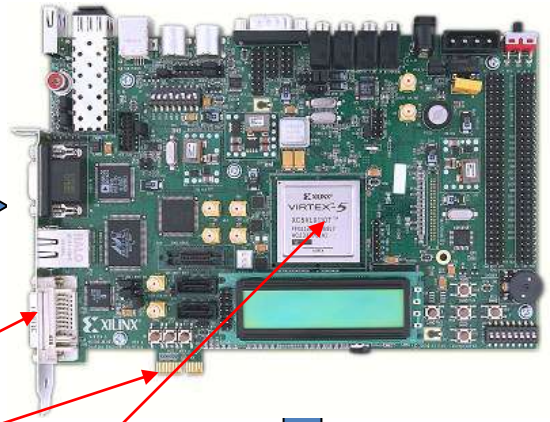
主要研究内容

1. 搭建医学图像三维重建系统框架
2. 实现计算机到FPGA PCIE医学图像数据传输
3. 实现FPGA平台医学图像硬件预处理
4. 实现医学图像DVI输出显示

主要研究内容

系统框架

PCI Express



DVI

驱动、应用程序

PCI Express 端口

RGB2Ycrcb、 Median filter、 Ycrcb2RGB

DVI控制芯片配置

1. 搭建医学图像三维重建系统框架
2. 实现计算机到FPGA PCIE医学图像数据传输
3. 实现FPGA平台医学图像硬件预处理
4. 实现医学图像DVI输出显示

Preliminaries

- PCI Express 简介

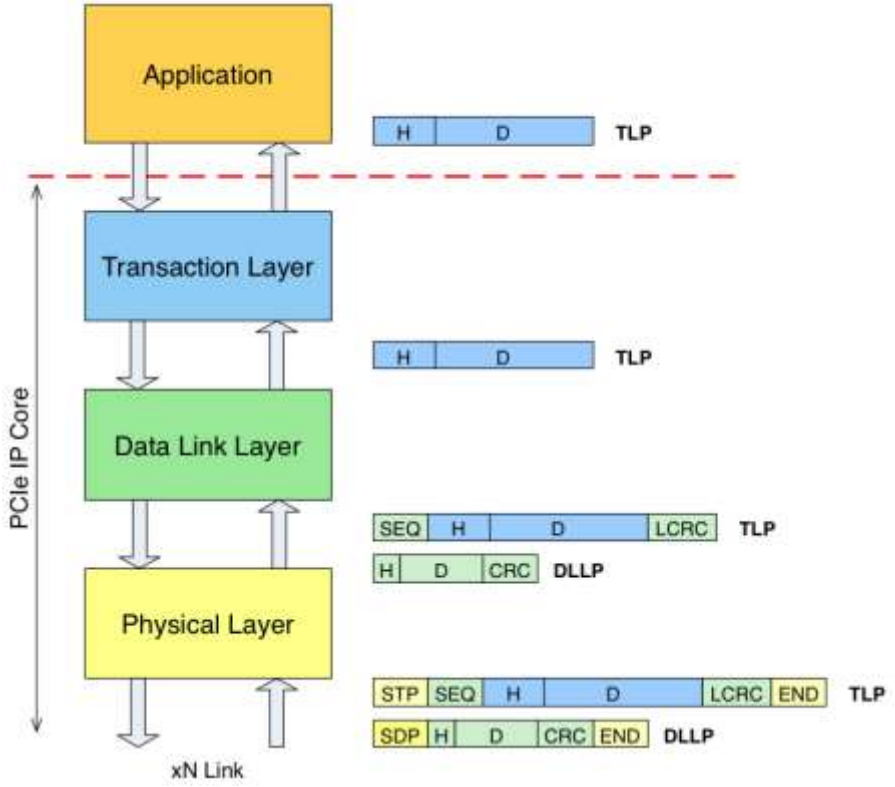
Protocols Supported	Data Rates	Lanes Supported
PCI Express (Rev 1.0a)	2.5 Gb/s	1, 4
XAUI 802.3ae D5p0	3.125 Gb/s	4
DisplayPort™	1.62 Gb/s, 2.7 Gb/s	1, 2, 4
EPON	1.25 Gb/s	1
GPON	622 Mb/s, 1.25 Gb/s, 2.5 Gb/s	1
Gigabit Ethernet	1.25 Gb/s	1
SATA Gen 1	1.5 Gb/s	1
CPRI (V3.0)	614.4 Mb/s, 1228.8 Mb/s, 2457.6 Mb/s, 3072.0 Mb/s	1
OBSAI RP3 (V4.0)	768 Mb/s, 1536 Mb/s, 3072 Mb/s	1
Serial RapidIO	1.25 Gb/s, 2.5 Gb/s, 3.125 Gb/s	1, 2, 4
Aurora	614 Mb/s - 3.2 Gb/s	1, 2, 4

Referred from Avnet

Preliminaries

- PCI Express 简介

- High-speed serial bus
- Layered architecture
- Application Data transferred via packets
- PCIe IP cores usually implement the lower
- Three layers
- PCIe IP cores solve most of the protocol handling



Preliminaries

- PCI Express 简介

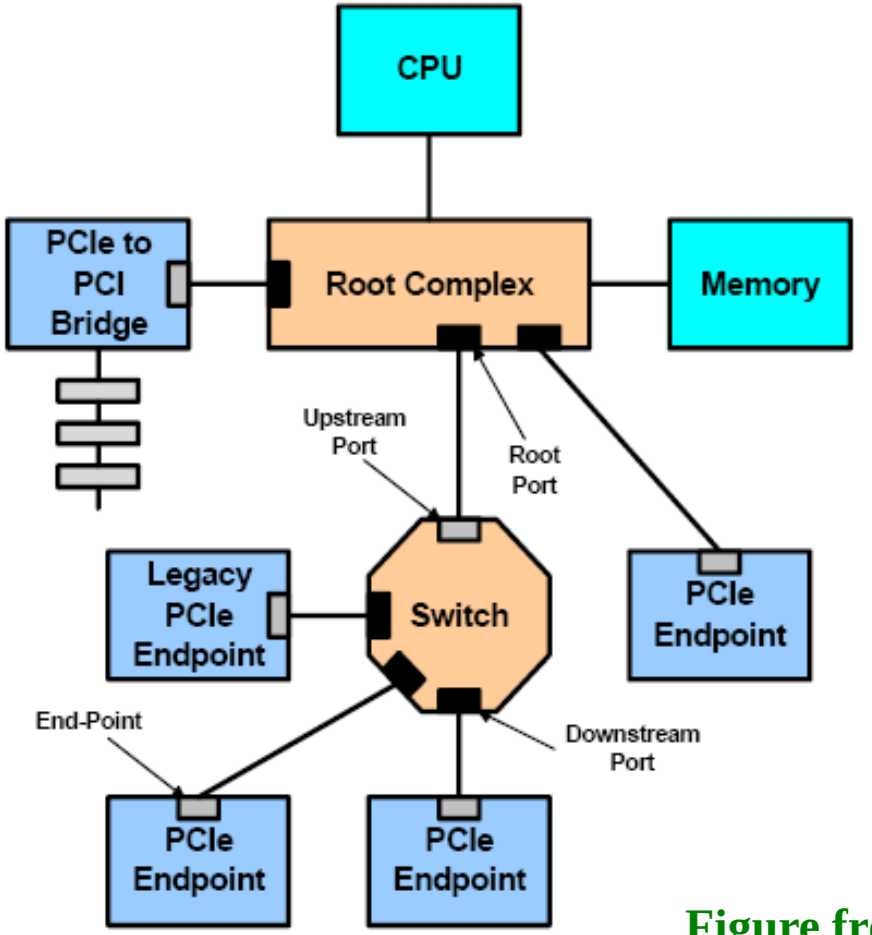


Figure from Avnet

Preliminaries

- PCI Express 简介

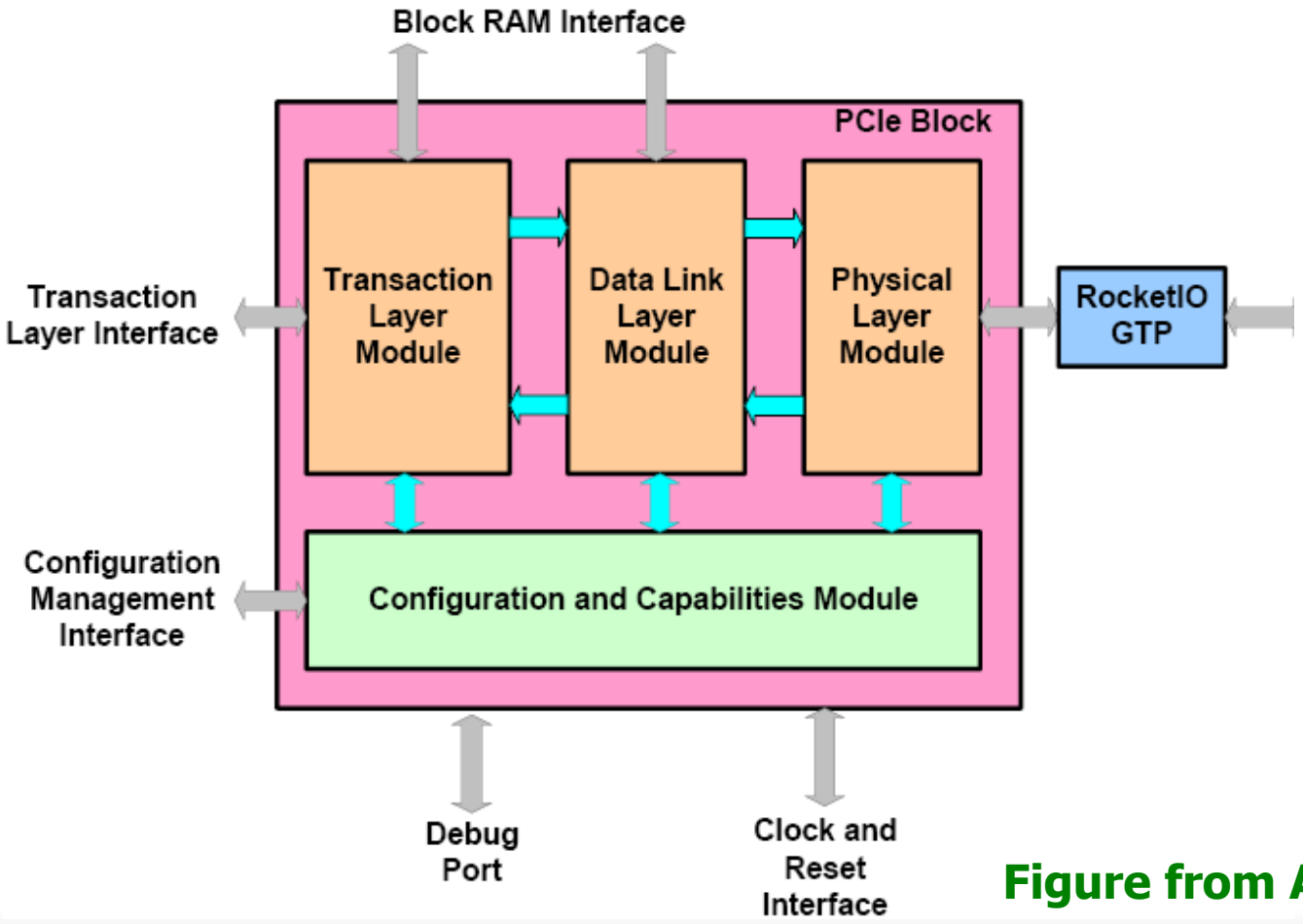


Figure from Avnet

Preliminaries

- PCI Express 简介

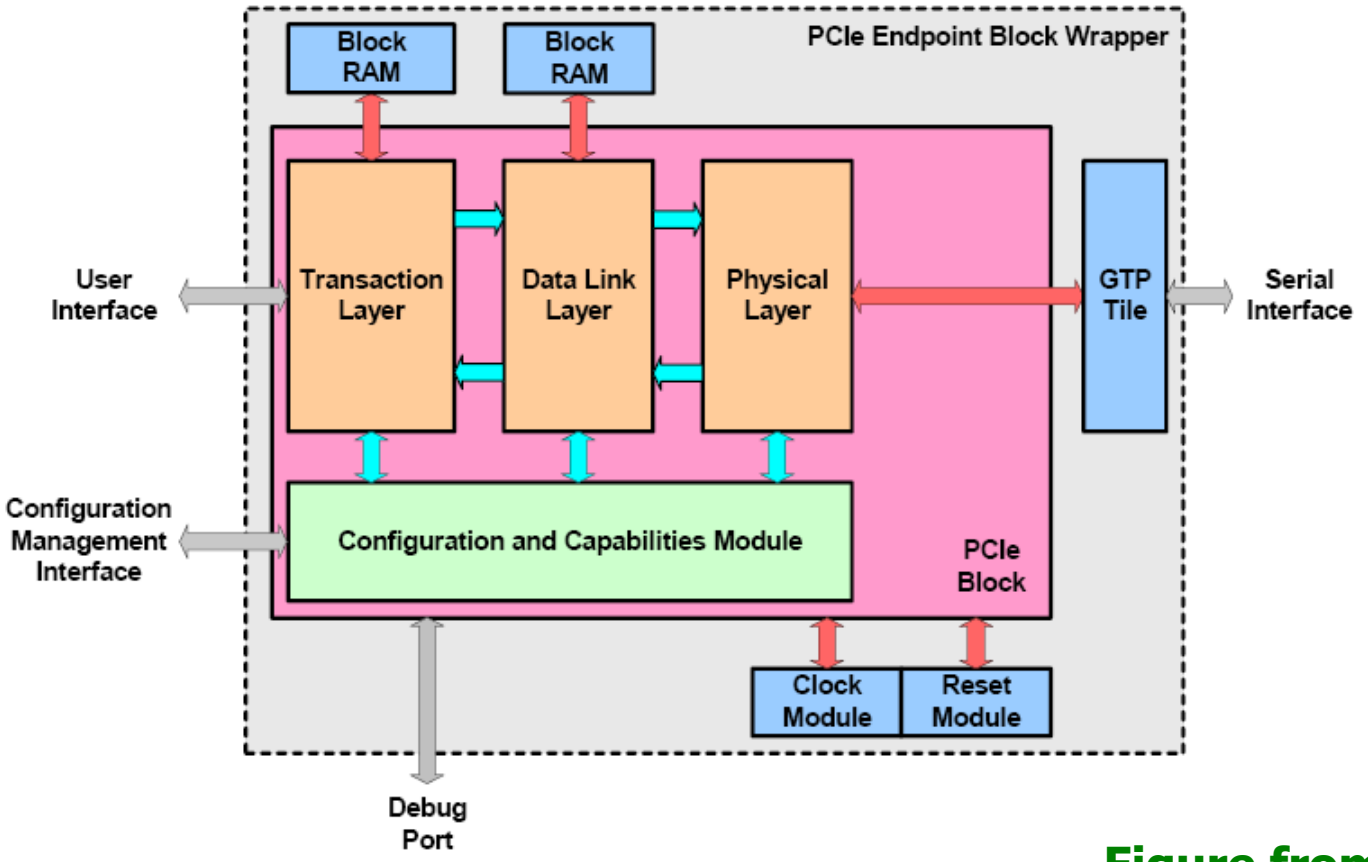
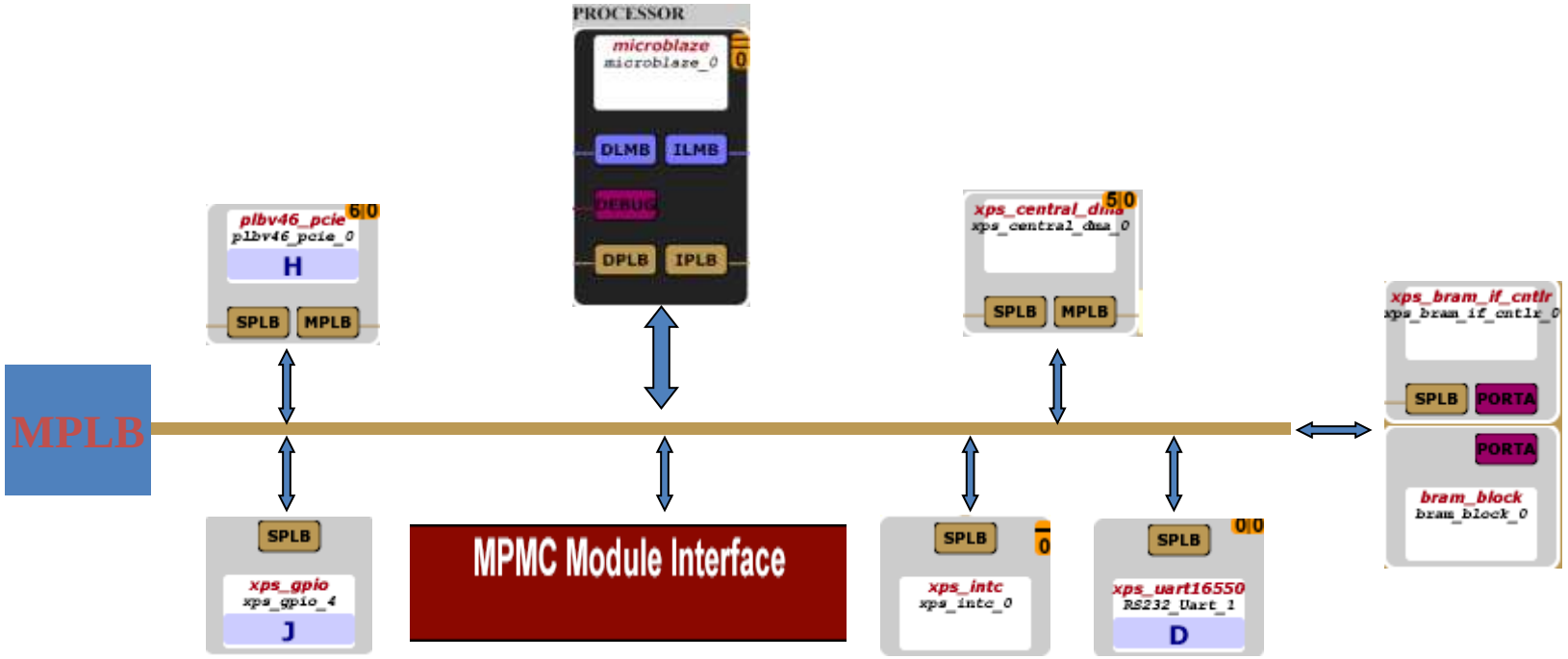


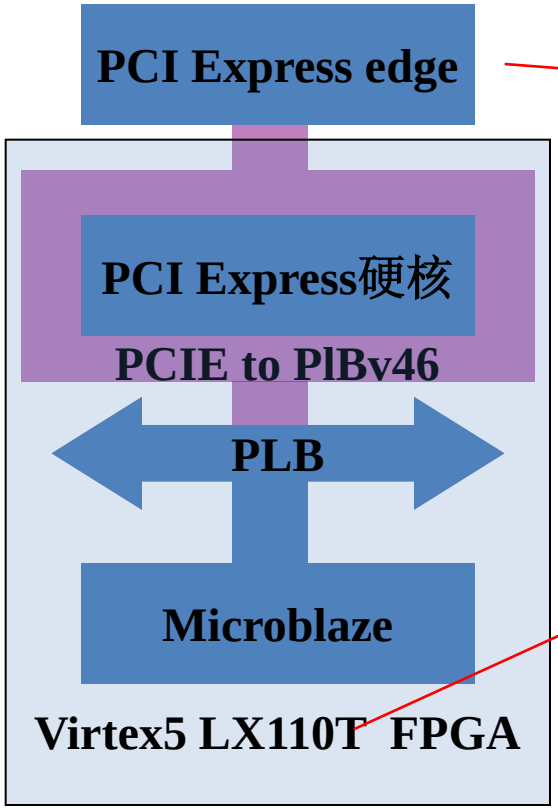
Figure from Avnet

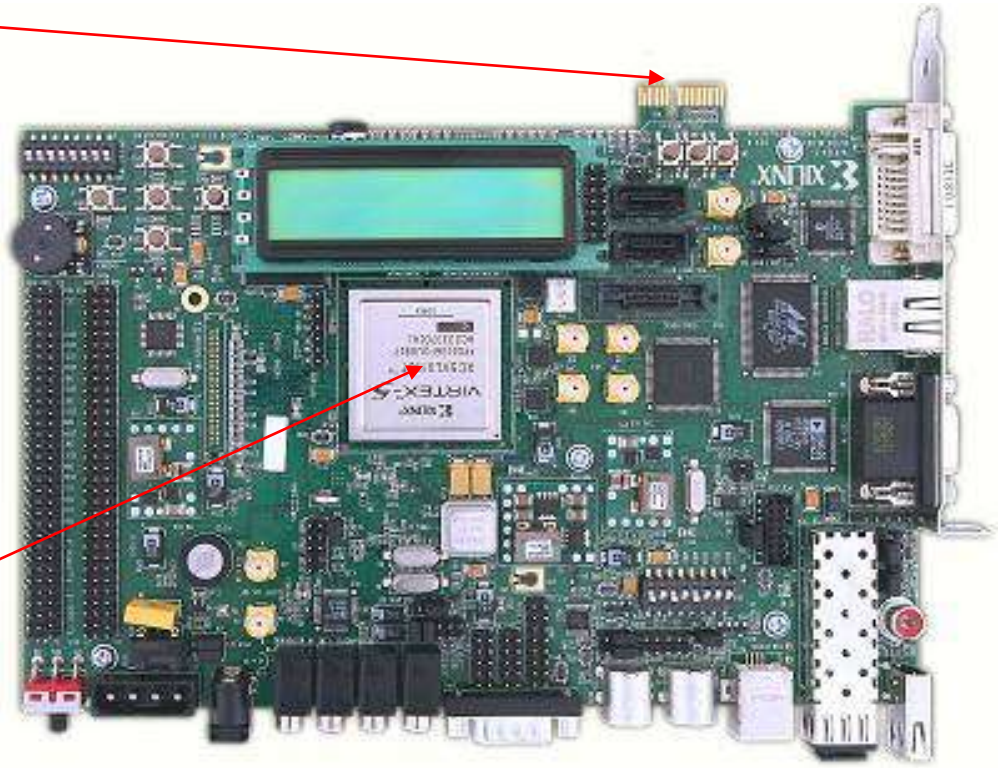
Preliminaries

- PCI Express数据端口设计

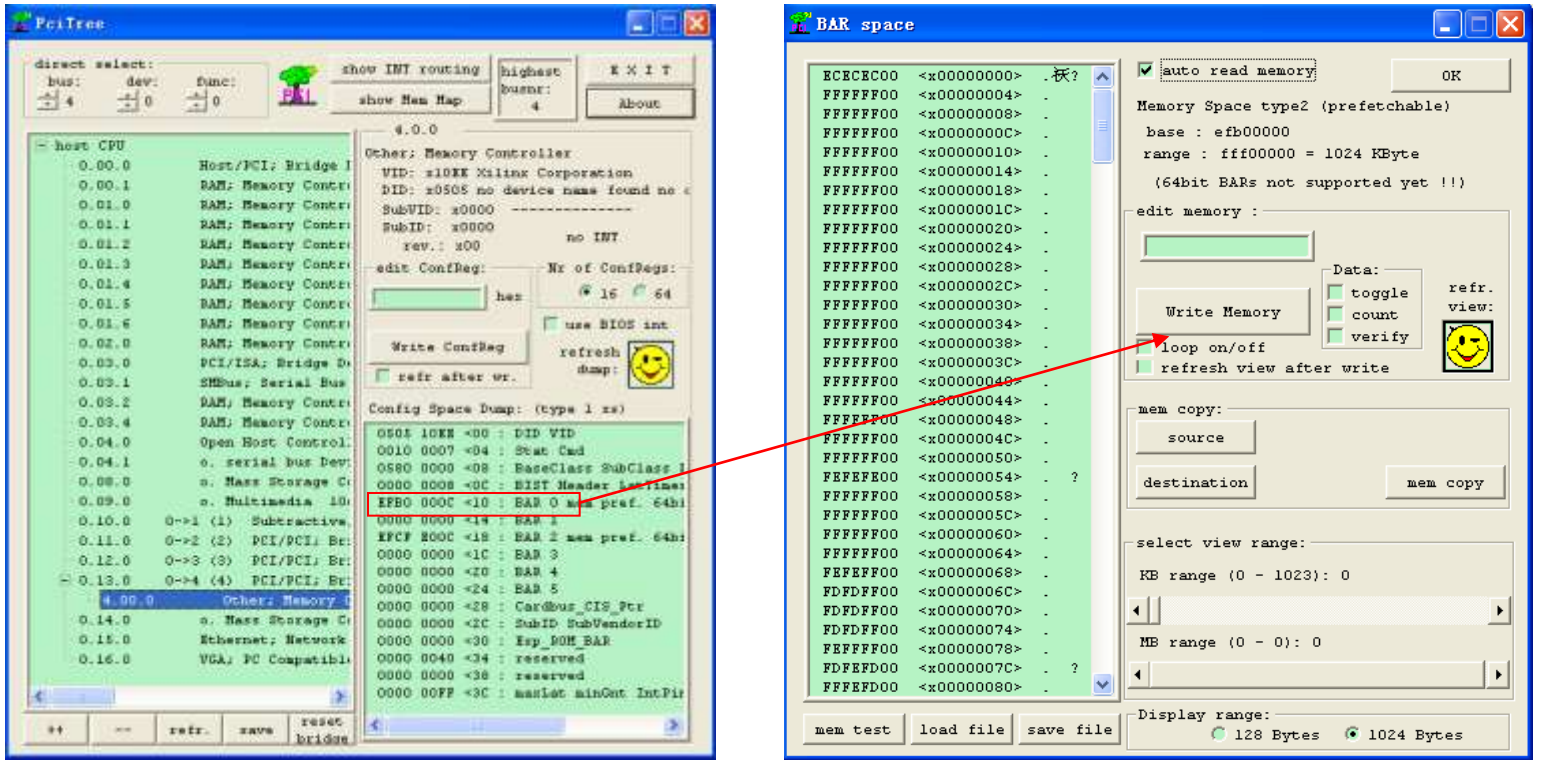


PCI Express数据端口设计

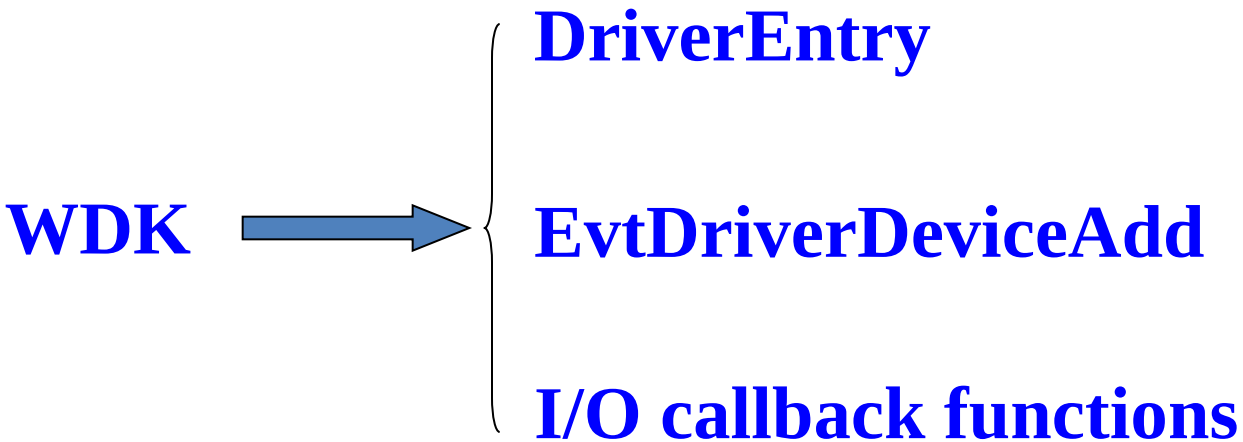




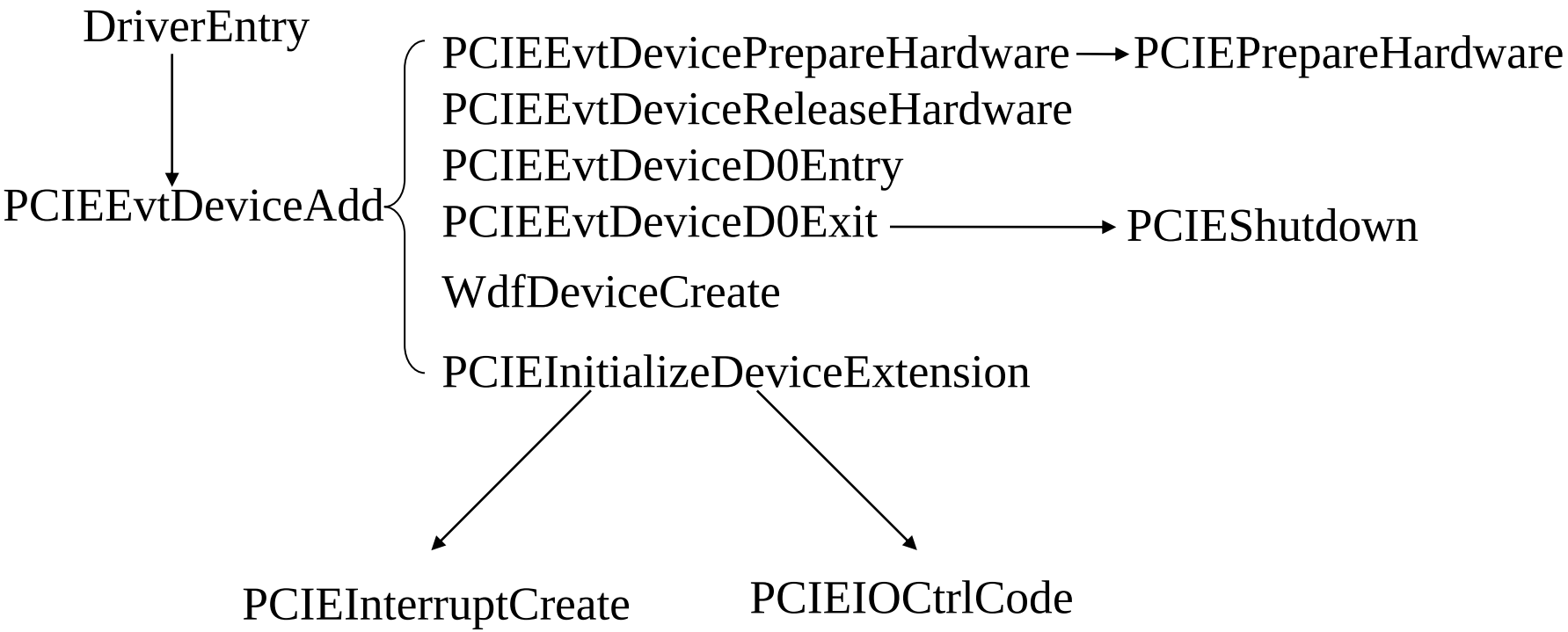
PCI Express数据端口验证



- 驱动框架


WDK
DriverEntry
EvtDriverDeviceAdd
I/O callback functions

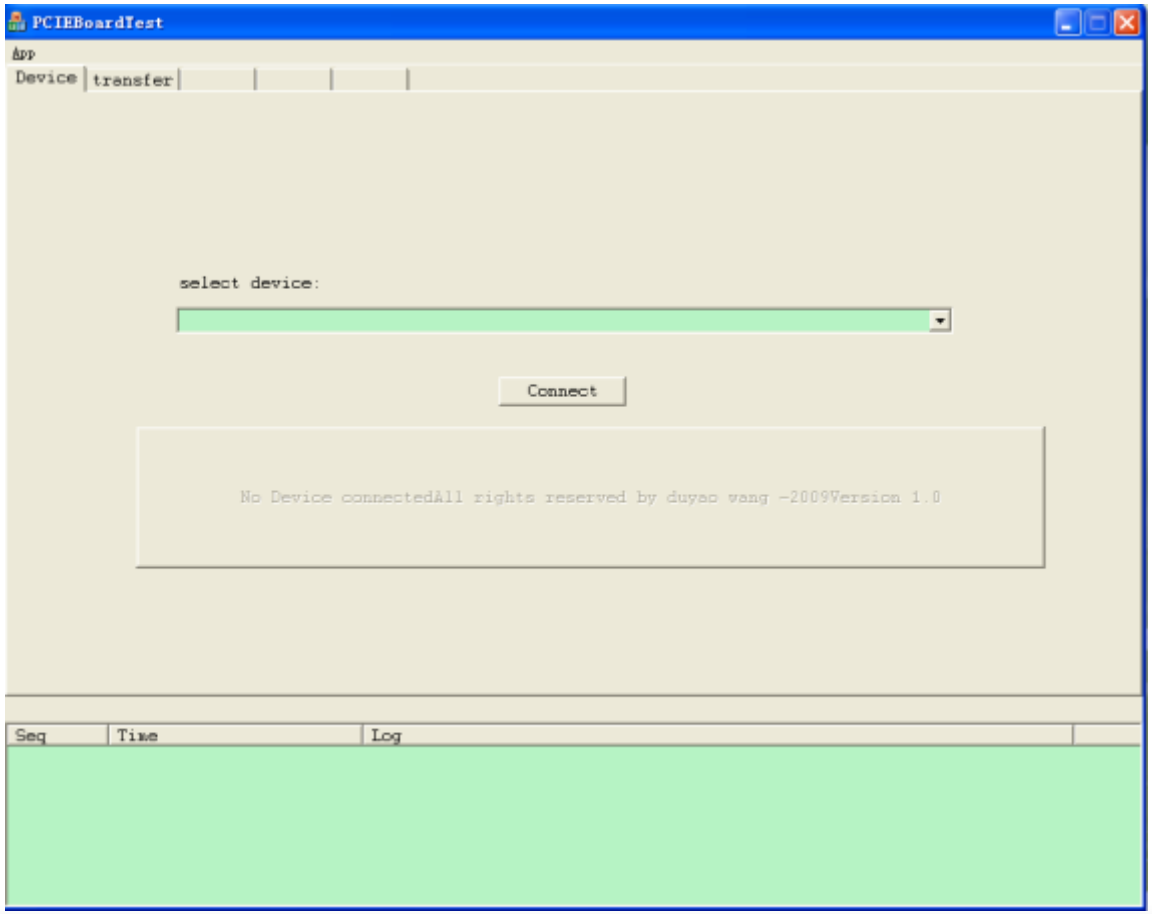
上位机驱动程序设计


DriverEntry
PCIEEvtDeviceAdd
PCIEEvtDevicePrepareHardware
PCIEPrepareHardware
PCIEEvtDeviceReleaseHardware
PCIEEvtDeviceD0Entry
PCIEEvtDeviceD0Exit
PCIEShutdown
WdfDeviceCreate
PCIEInitializeDeviceExtension
PCIEInterruptCreate
PCIEIOCtrlCode

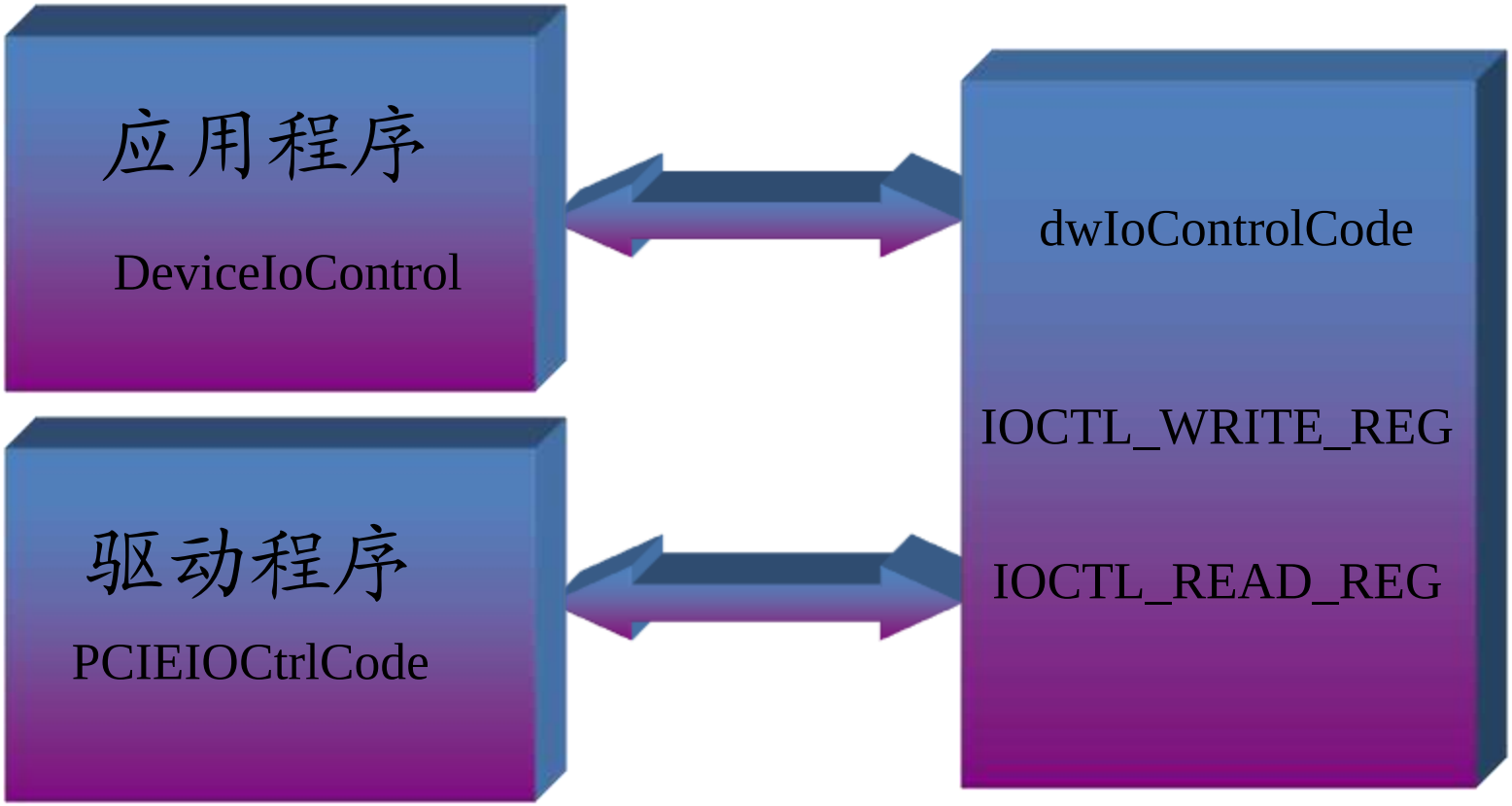
驱动、应用程序通信

```
BOOL DeviceIoControl (
  HANDLE hDevice,  // CreateFile返回的设备句柄
  DWORD dwIoControlCode, //应用程序调用驱动程序的控制命令
  LPVOID lpInBuffer, //应用程序传递给驱动程序的数据缓冲区地址
  DWORD nInBufferSize, //应用程序传递给驱动程序的数据缓冲区大小，字节数
  LPVOID lpOutBuffer, //驱动程序返回给应用程序的数据缓冲区地址
  DWORD nOutBufferSize, //驱动程序返回给应用程序的数据缓冲区大小，字节数
  LPDWORD lpBytesReturned, //驱动程序实际返回给应用程序的数据字节数地址
  LPOVERLAPPED lpOverlapped//重叠操作结构
);
```

- 应用程序界面

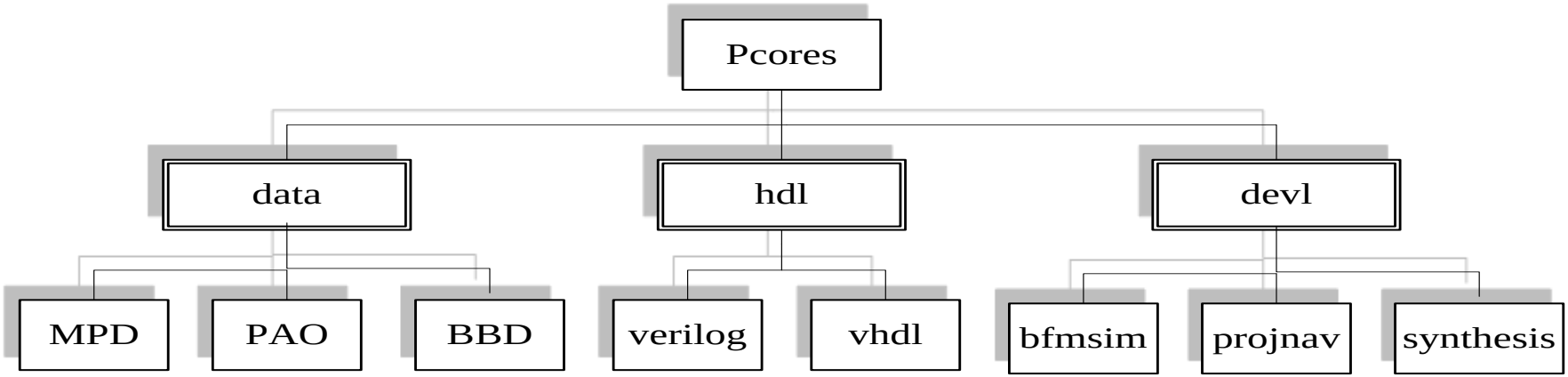

PCIEBoardTest
App
Device
transfer
select device:
Connect
No Device connectedAll rights reserved by duyao wang -2009Version 1.0
Seq
Time
Log

- 驱动、应用程序通信

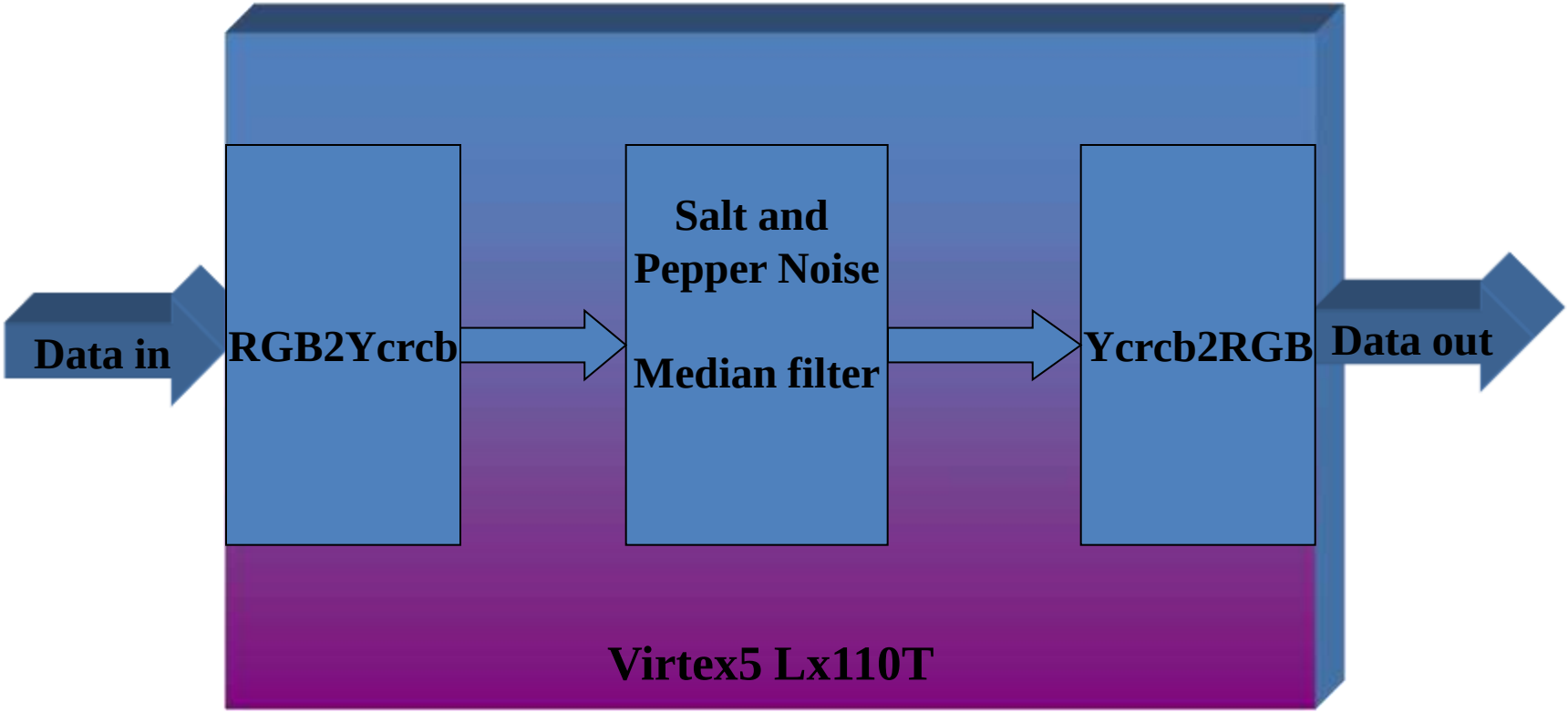

应用程序
DeviceIoControl
驱动程序
PCIEIOCtrlCode
dwIoControlCode
IOCTL_WRITE_REG
IOCTL_READ_REG

1. 搭建医学图像三维重建系统框架
2. 实现计算机到FPGA PCIE医学图像数据传输
3. 实现FPGA平台医学图像硬件预处理
4. 实现医学图像DVI输出显示

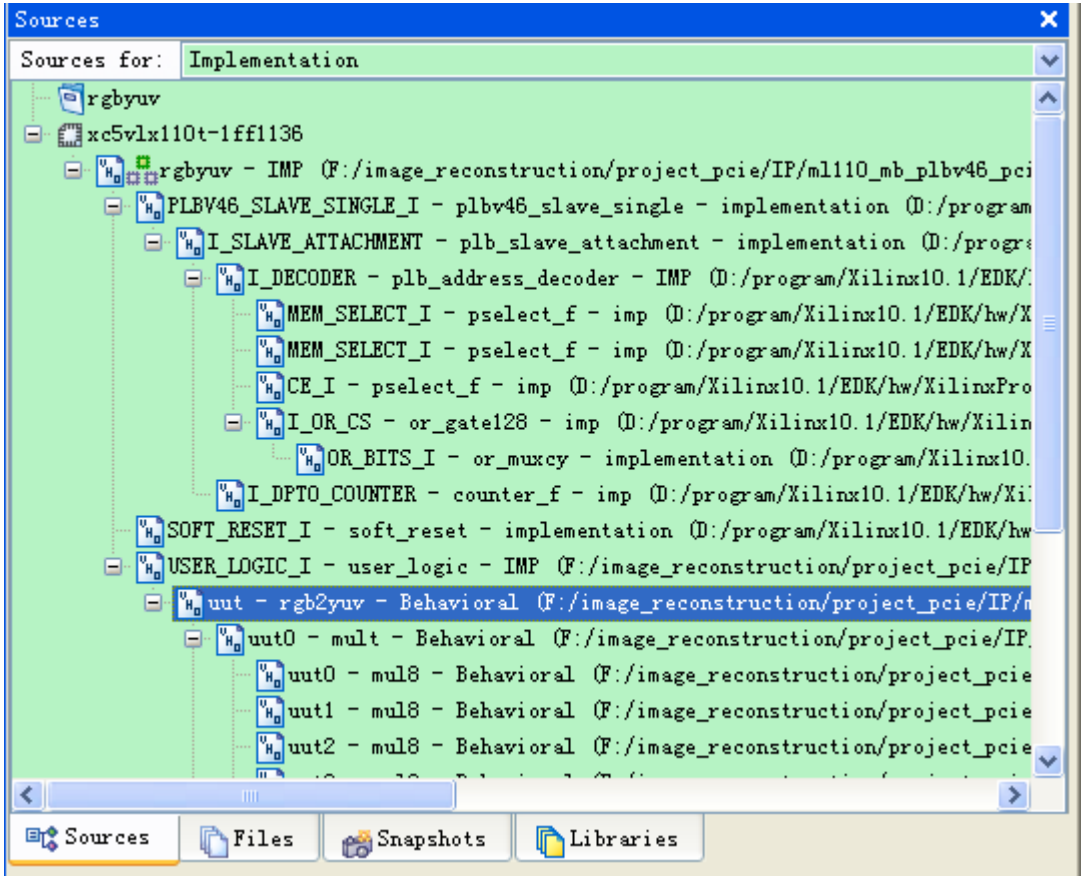
- 自定义医学图像处理IP定制



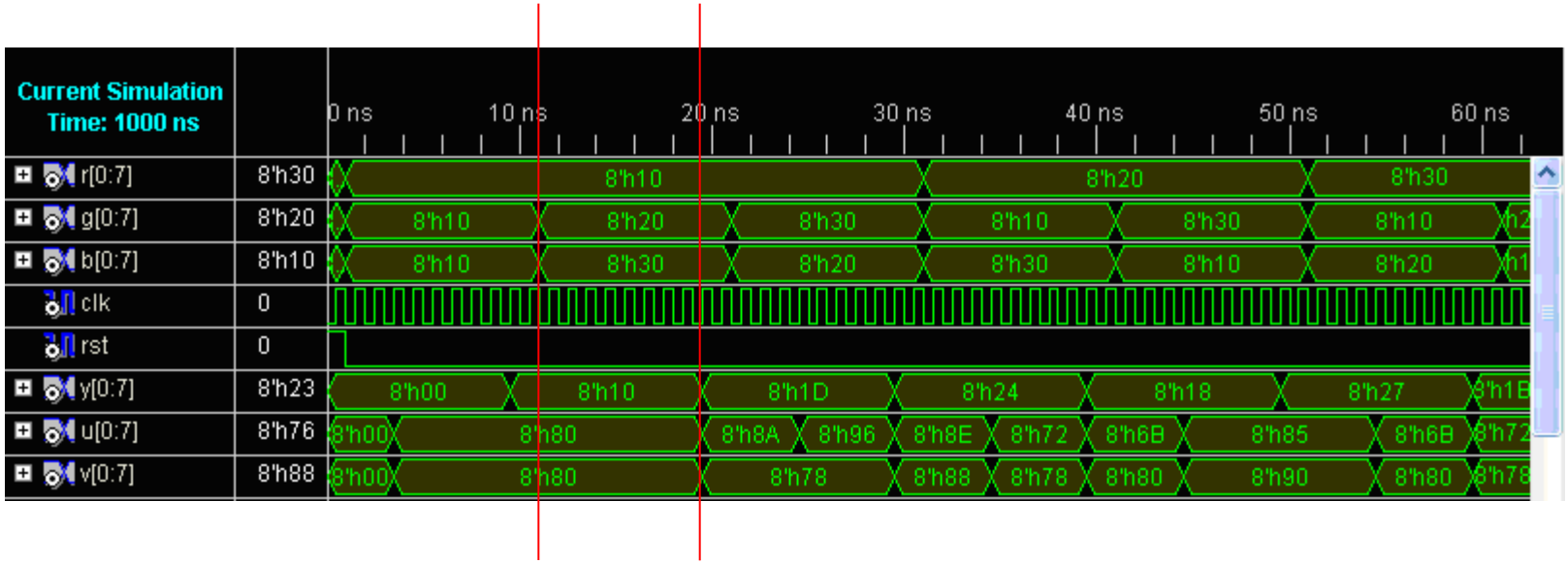
- 自定义医学图像处理IP定制



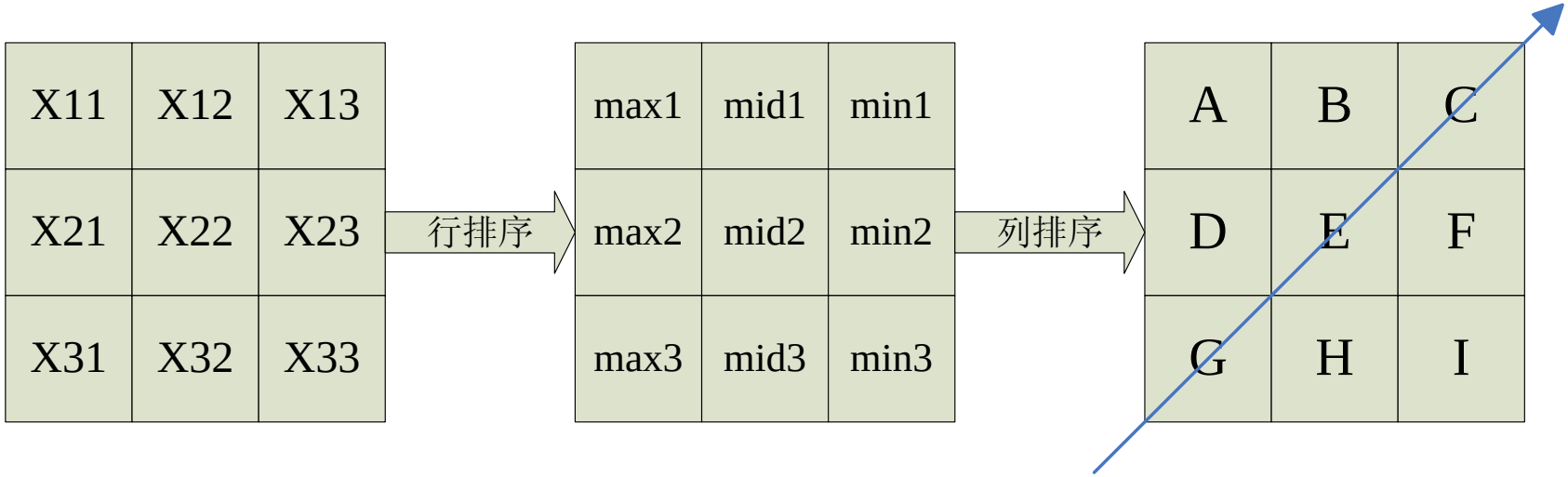
- **RGB2Ycrcb**


Sources
Sources for: Implementation
rgbyuv
xc5vlx110t-1ff1136
rgbyuv - IMP (F:/image_reconstruction/project_pcie/IP/ml110_mb_plbv46_pci
PLBV46_SLAVE_SINGLE_I - plbv46_slave_single - implementation (D:/program
I_SLAVE_ATTACHMENT - plb_slave_attachment - implementation (D:/progr
I_DECODER - plb_address_decoder - IMP (D:/program/Xilinx10.1/EDK/
MEM_SELECT_I - pselect_f - imp (D:/program/Xilinx10.1/EDK/hw/X
MEM_SELECT_I - pselect_f - imp (D:/program/Xilinx10.1/EDK/hw/X
CE_I - pselect_f - imp (D:/program/Xilinx10.1/EDK/hw/XilinxPro
I_OR_CS - or_gate128 - imp (D:/program/Xilinx10.1/EDK/hw/Xilin
OR_BITS_I - or_muxcy - implementation (D:/program/Xilinx10.
I_DPTO_COUNTER - counter_f - imp (D:/program/Xilinx10.1/EDK/hw/Xi
SOFT_RESET_I - soft_reset - implementation (D:/program/Xilinx10.1/EDK/hw
USER_LOGIC_I - user_logic - IMP (F:/image_reconstruction/project_pcie/IP
uut - rgb2yuv - Behavioral (F:/image_reconstruction/project_pcie/IP/
uut0 - mult - Behavioral (F:/image_reconstruction/project_pcie/IP
uut0 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut1 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut2 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
Sources
Files
Snapshots
Libraries

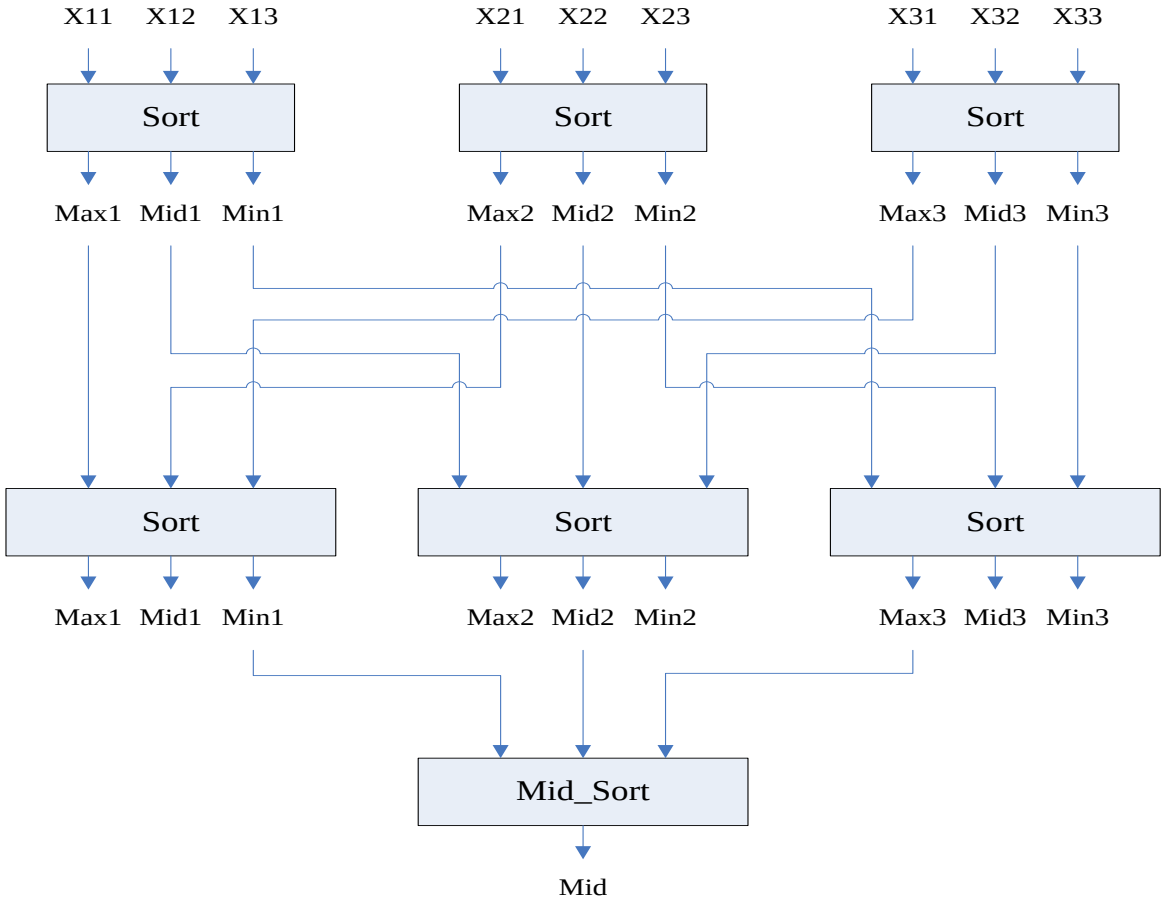
- **RGB2Ycrcb**


Current Simulation Time: 1000 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
r[0:7] 8'h30 8'h10 8'h20 8'h30
g[0:7] 8'h20 8'h10 8'h20 8'h30 8'h10 8'h30 8'h10
b[0:7] 8'h10 8'h10 8'h30 8'h20 8'h30 8'h10 8'h20
clk 0
rst 0
y[0:7] 8'h23 8'h00 8'h10 8'h1D 8'h24 8'h18 8'h27
u[0:7] 8'h76 8'h00 8'h80 8'h8A 8'h96 8'h8E 8'h72 8'h6B 8'h85 8'h6B
v[0:7] 8'h88 8'h00 8'h80 8'h78 8'h88 8'h78 8'h80 8'h90 8'h80

- 快速中值滤波算法


X11 X12 X13
X21 X22 X23
X31 X32 X33
行排序
max1 mid1 min1
max2 mid2 min2
max3 mid3 min3
列排序
A B C
D E F
G H I

- 快速中值滤波VHDL实现



快速中值滤波VHDL实现

```
uutsort1: sort PORT MAP (
        slv_regin1 => slv_reg0,
        slv_regin2 => slv_reg1,
        slv_regin3 => slv_reg2,
       slv_regout1 => slv_regtmp0,
        slv_regout2 => slv_regtmp1,
        slv_regout3 => slv_regtmp2,
        Bus2IP_Clk => Bus2IP_Clk );

uutsort2: sort PORT MAP (
        slv_regin1 => slv_reg3,
        slv_regin2 => slv_reg4,
        slv_regin3 => slv_reg5,
        slv_regout1 => slv_regtmp3,
        slv_regout2 => slv_regtmp4,
        slv_regout3 => slv_regtmp5,
        Bus2IP_Clk => Bus2IP_Clk );
```

- # 快速中值滤波VHDL实现

```
uutsort3: sort PORT MAP (
        slv_regin1 => slv_reg6,
        slv_regin2 => slv_reg7,
        slv_regin3 => slv_reg8,
        slv_regout1 => slv_regtmp6,
        slv_regout2 => slv_regtmp7,
        slv_regout3 => slv_regtmp8,
        Bus2IP_Clk => Bus2IP_Clk
                                );
uutsort4: sort PORT MAP (
        slv_regin1 =>  slv_regtmp0,
        slv_regin2 =>  slv_regtmp3,
        slv_regin3 =>  slv_regtmp6,
        slv_regout1 => slv_regtmpout0,
        slv_regout2 => slv_regtmpout3,
        slv_regout3 => slv_regtmpout6,
        Bus2IP_Clk  => Bus2IP_Clk
                                );
```

快速中值滤波VHDL实现

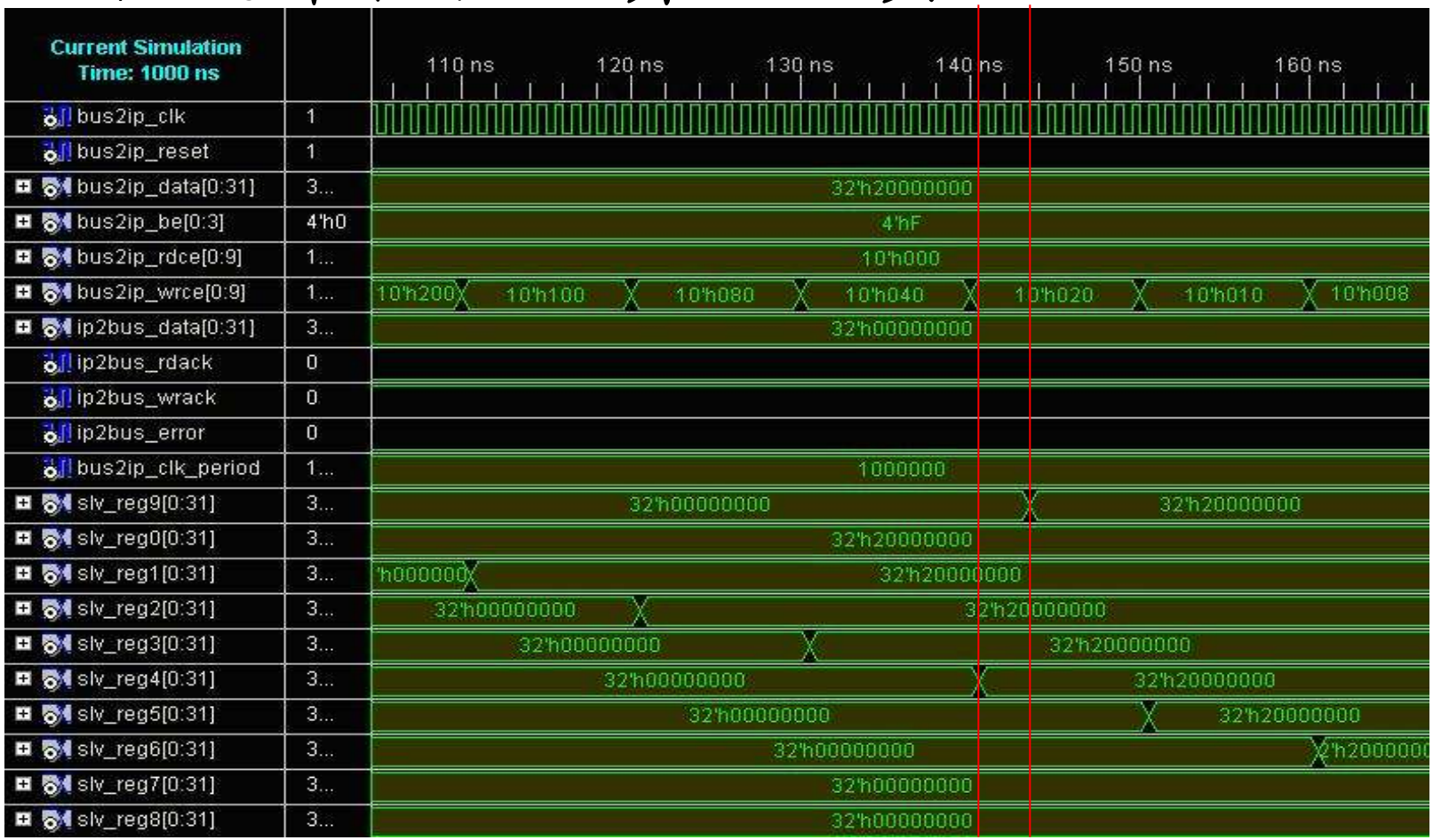
```
uutsort5: sort PORT MAP (
        slv_regin1 =>  slv_regtmp1,
        slv_regin2 =>  slv_regtmp4,
        slv_regin3 =>  slv_regtmp7,
              slv_regout1 => slv_regtmpout1,
        slv_regout2 => slv_regtmpout4,
        slv_regout3 => slv_regtmpout7,
        Bus2IP_Clk  => Bus2IP_Clk );

uutsort6: sort PORT MAP (
        slv_regin1 =>  slv_regtmp2,
        slv_regin2 =>  slv_regtmp5,
        slv_regin3 =>  slv_regtmp8,
              slv_regout1 => slv_regtmpout2,
        slv_regout2 => slv_regtmpout5,
        slv_regout3 => slv_regtmpout8,
        Bus2IP_Clk  => Bus2IP_Clk );
```

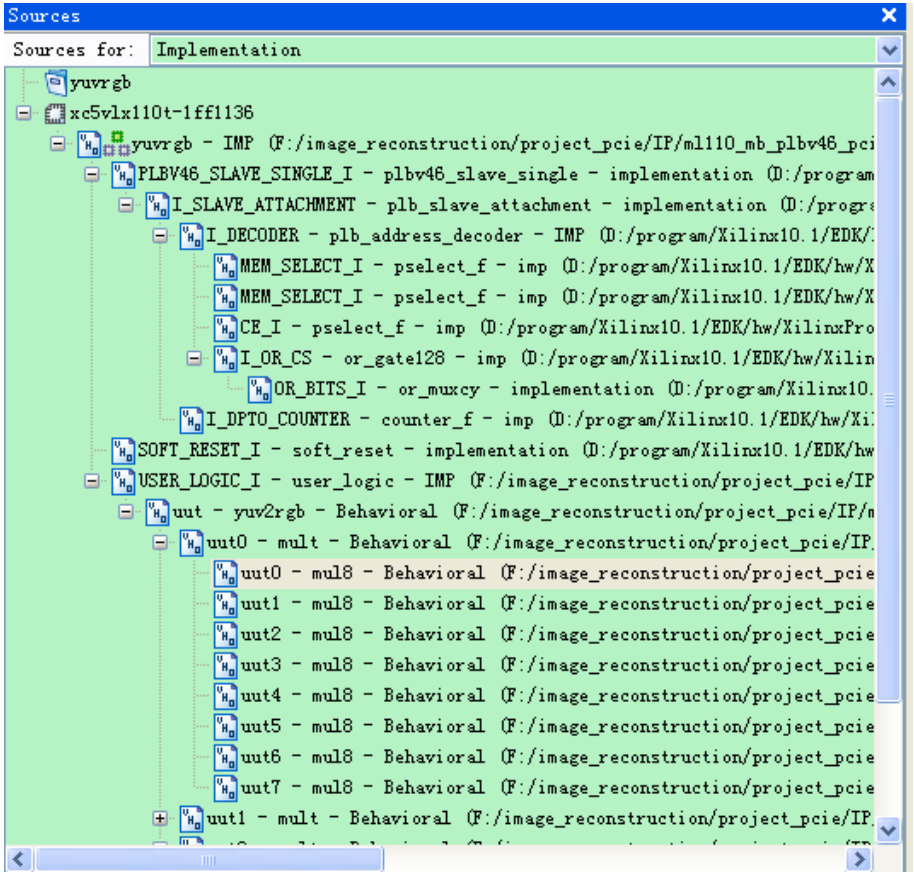
- 快速中值滤波VHDL实现

```
uutsort7: sort_mid PORT MAP (
        slv_regin1 => slv_regtmpout2,
        slv_regin2 => slv_regtmpout4,
        slv_regin3 => slv_regtmpout6,
        slv_regout => slv_reg9,
        Bus2IP_Clk => Bus2IP_Clk );
```

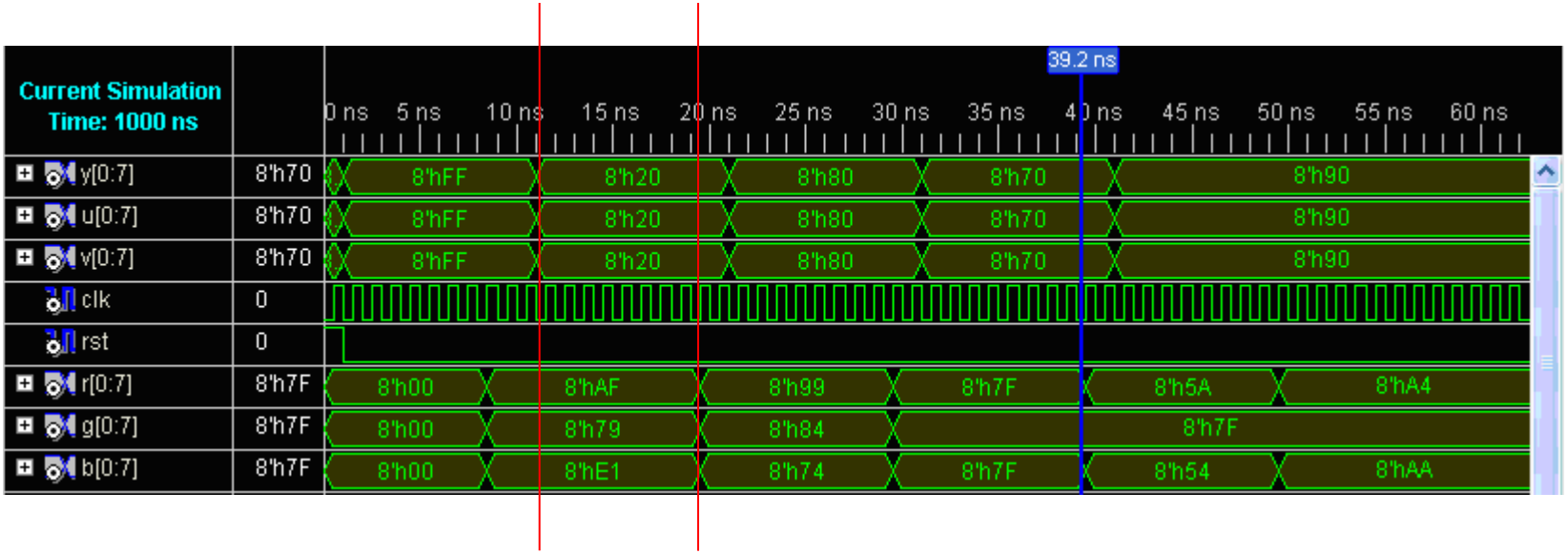
- 快速中值濾波算法仿真


Current Simulation Time: 1000 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
bus2ip_clk
bus2ip_reset
bus2ip_data[0:31]
bus2ip_be[0:3]
bus2ip_rdce[0:9]
bus2ip_wrce[0:9]
ip2bus_data[0:31]
ip2bus_rdack
ip2bus_wrack
ip2bus_error
bus2ip_clk_period
slv_reg9[0:31]
slv_reg0[0:31]
slv_reg1[0:31]
slv_reg2[0:31]
slv_reg3[0:31]
slv_reg4[0:31]
slv_reg5[0:31]
slv_reg6[0:31]
slv_reg7[0:31]
slv_reg8[0:31]
32'h20000000
4'hF
10'h000
10'h200
10'h100
10'h080
10'h040
10'h020
10'h010
10'h008
32'h00000000
1000000

- **Ycrcb2RGB**

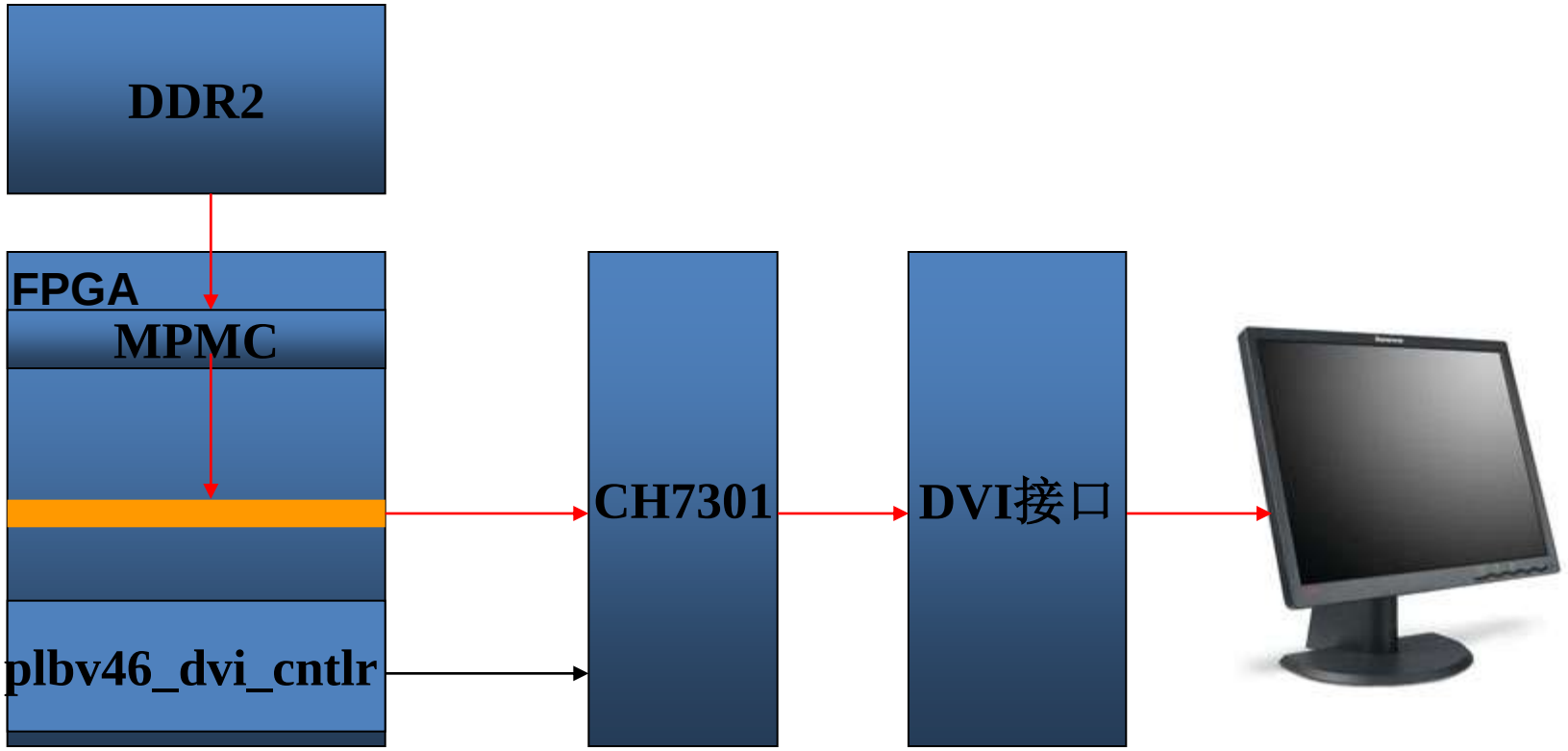

Sources
Sources for: Implementation
yuvrgb
xc5vlx110t-1ff1136
yuvrgb - IMP (F:/image_reconstruction/project_pcie/IP/ml110_mb_plbv46_pci
PLBV46_SLAVE_SINGLE_I - plbv46_slave_single - implementation (D:/program
I_SLAVE_ATTACHMENT - plb_slave_attachment - implementation (D:/progr
I_DECODER - plb_address_decoder - IMP (D:/program/Xilinx10.1/EDK/
MEM_SELECT_I - pselect_f - imp (D:/program/Xilinx10.1/EDK/hw/X
MEM_SELECT_I - pselect_f - imp (D:/program/Xilinx10.1/EDK/hw/X
CE_I - pselect_f - imp (D:/program/Xilinx10.1/EDK/hw/XilinxPro
I_OR_CS - or_gate128 - imp (D:/program/Xilinx10.1/EDK/hw/Xilin
OR_BITS_I - or_muxcy - implementation (D:/program/Xilinx10.
I_DPTO_COUNTER - counter_f - imp (D:/program/Xilinx10.1/EDK/hw/Xi
SOFT_RESET_I - soft_reset - implementation (D:/program/Xilinx10.1/EDK/hw
USER_LOGIC_I - user_logic - IMP (F:/image_reconstruction/project_pcie/IP
uut - yuv2rgb - Behavioral (F:/image_reconstruction/project_pcie/IP/
uut0 - mult - Behavioral (F:/image_reconstruction/project_pcie/IP.
uut0 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut1 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut2 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut3 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut4 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut5 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut6 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut7 - mul8 - Behavioral (F:/image_reconstruction/project_pcie
uut1 - mult - Behavioral (F:/image_reconstruction/project_pcie/IP.

- **Ycrcb2RGB**


Current Simulation Time: 1000 ns
39.2 ns
0 ns 5 ns 10 ns 15 ns 20 ns 25 ns 30 ns 35 ns 40 ns 45 ns 50 ns 55 ns 60 ns
y[0:7] 8'h70 8'hFF 8'h20 8'h80 8'h70 8'h90
u[0:7] 8'h70 8'hFF 8'h20 8'h80 8'h70 8'h90
v[0:7] 8'h70 8'hFF 8'h20 8'h80 8'h70 8'h90
clk 0
rst 0
r[0:7] 8'h7F 8'h00 8'hAF 8'h99 8'h7F 8'h5A 8'hA4
g[0:7] 8'h7F 8'h00 8'h79 8'h84 8'h7F
b[0:7] 8'h7F 8'h00 8'hE1 8'h74 8'h7F 8'h54 8'hAA

1. 搭建医学图像三维重建系统框架
2. 实现计算机到FPGA PCIE医学图像数据传输
3. 实现FPGA平台医学图像硬件预处理
4. 实现医学图像DVI输出显示

DVI输出显示



- **FPGA硬件滤波处理结果**

- 系统图

存储系统

处理系统

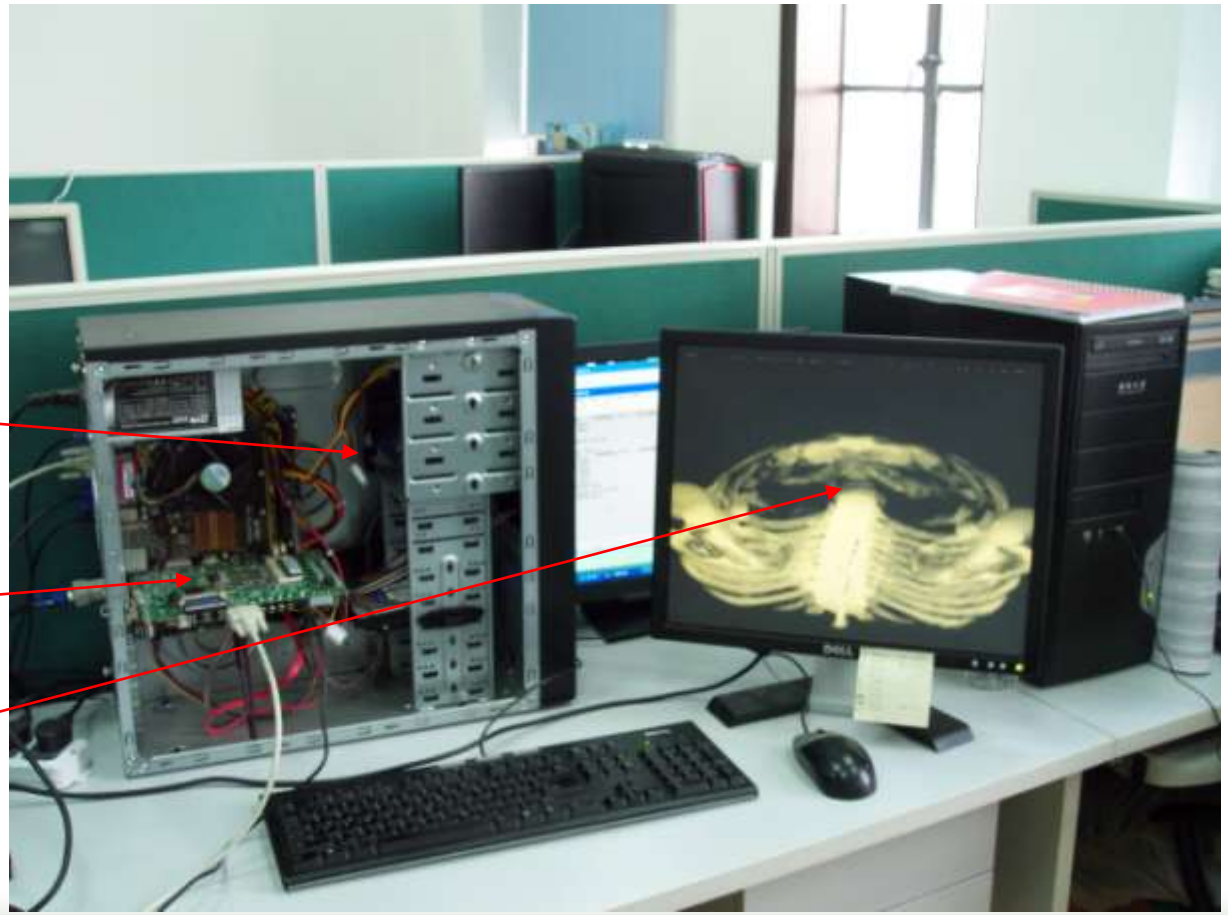
显示系统

- 系统图

存储系统

处理系统

显示系统



FPGA资源利用率

使用资源	使用数	百分比
Number of Slice Registers	19116 out of 69120	27%
Number of Slice LUTS	20864 out of 69120	30%
Number of Slice LUT-Flip Flop pairs	27990 out of 69120	40%
Number of External IOBs	186 out of 640	29%
Number of BUFGs	13 out of 32	40%

主要内容

研究背景、意义

国内外研究现状

主要研究内容

下一步工作

总结

下一步工作

1. 支持DMA传输
   - 1.1 DDK/WKD
   - 1.2 Win driver
2. 实现三维重建算法移植

谢谢！