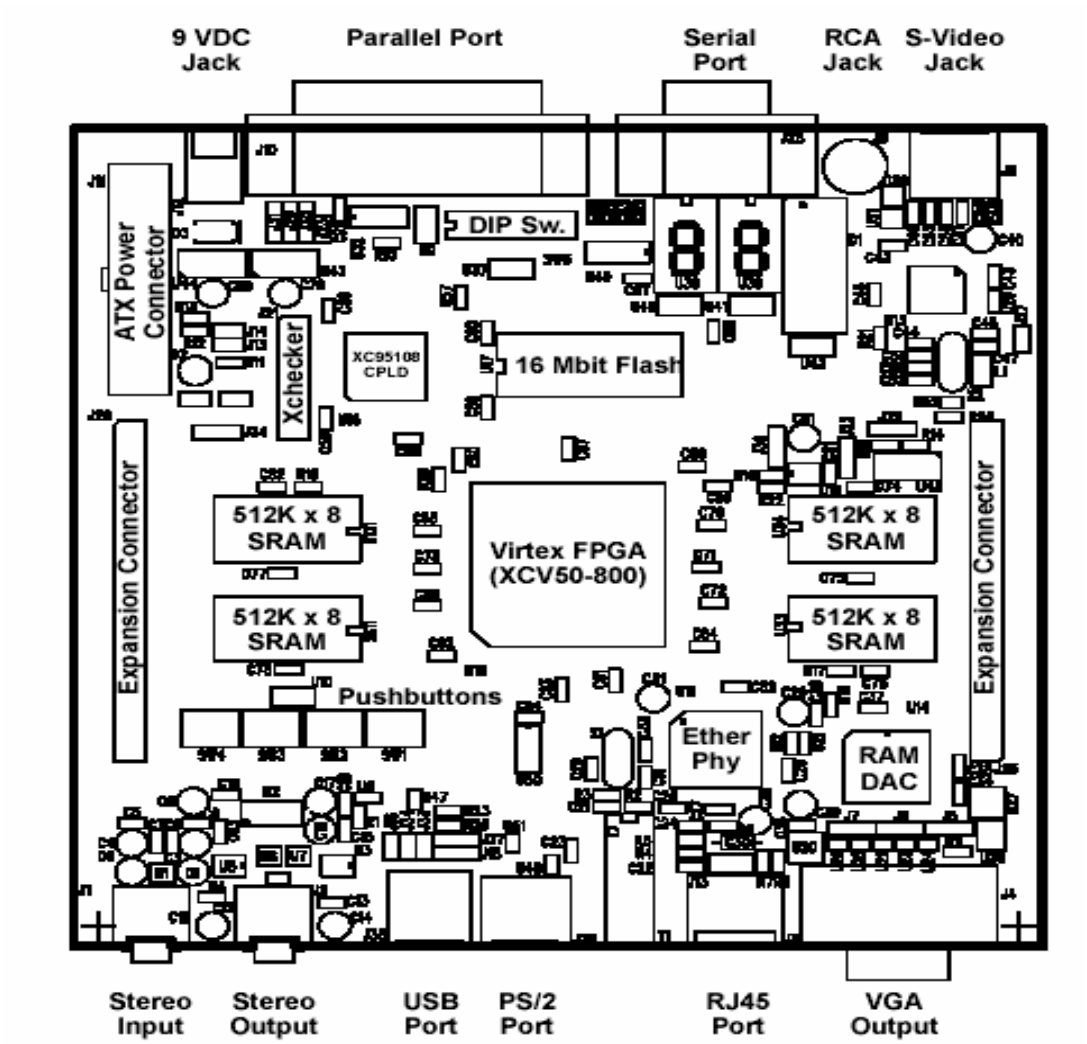
第一章　XSV800 Board 简介

XSV800 Board 提供各种各样的接口与外界进行通讯：串口，并口，Xchecker 电缆接口，USB 接口，PS/2 的鼠标键盘接口，10/100 以太网物理层接口。而主体部分是一块 Virtex FPGA(XSV800)。一般通过串口，并口，Xchecker 电缆来配置 Virtex FPGA，也可以通过存储在 16M Flash RAM 中的 Bitstream 来配置。测试板的结构图如下**(图 1.1)**：



(图 1.1)

XSV800 Board 拥有的可用资源包括：

} Xilinx Virtex FPGA：Xilinx FPGA 从 57K Gates(XSV50)到 888K Gates(XSV800)，它们都是采用 240PIN 的 PQFP 或 HQFP 封装。这里是个 888K Gates 的 FPGA，它是这块 XSV800 Board 的可编程逻辑的主要存储库。

} Xilinx XC95108 CPLD：用于通过串口，并口，或 Flash RAM 来配置 XSV800 Board，同时也控制着以太网芯片的配置。

} 可编程振荡器(Programmable Oscillator)：以 100MHz 的基准频率为源向 FPGA 和 CPLD 提供时钟信号。

} 16M Flash RAM：存储 FPGA 所用配置信号和多用途的数据。

} 两组独立的 512K*16Bits SRAM：FPGA 存储所要用的数据。

} 视频解码器(Video Decoder)：通过 RCA 插孔或 SVideo 连接器接受 NTSC/PAL/SECAM 信号并且向 FPGA 输出数字化的视频信号。

} RAMDAC：提供 FPGA 向显示器输出 VGA 信号的色彩表支持。

} 立体声多媒体数字信号编码器(Stereo codec)：数字化并产生 0—50KHz 的音频信号。

} 10/100 的以太网芯片(10BASE-T/100BASE-TX Ethernet PHY Chip)：FPGA 用它来在 LAN 上以最高 100 Mbps 接受或发送数据。

} 四按钮和一组 DIP 开关：向 FPGA,CPLD 提供多功能输入。

} 3 根 LED：显示 FPGA,CPLD 的状态信息。

} PS/2 鼠标键盘接口(Mouse/Keyboard PS/2 Port)：让 FPGA 能够使用通用的输入设备。

} USB 接口(USB port)：向 FPGA 提供带宽为 1.5 to12 Mbps 的 I/O 信道。

} 串/并连接口(Parallel/Serial Port)：可以让 CPLD 像 PC 一样接受，发送串/并格式的数据。

} Xchecker 电缆接口(Xchecker Cable Interface)：下载配置或者读回配置。

ATX 电源的连接器或者 9V 的直流电源插孔使得 XSV800 Board 可以接受标准的 ATX 电源或 9V 的直流电源。

第二章 视频信号输入处理

本章介绍系统处理全国电视委员会制式(NTSC)或顺序与存储彩色电视系统制式(SECAM)或逐行倒相制式(PAL)的视频信号的方案。

¿2.1 概述

在Active HDL上创建一个新的项目，实验中我将其命名为VideoIn，用以对FPGA编程，使其完成从特定的接口接受NTSC, SECAM或PAL制式的视频信号，并将其数字化储存在测试板上的缓冲区(两块独立的512K x 16 SRAM)中的工作。上述程序可以通过使用VHDL或Verilog来描述，这次实验中使用的是VHDL。这个程序的最上层是一个名为saa7113的模块，且包含了以下子模块：sram512kleft16bit50mhz-sv01b(用来将数字化后视频信号数据存入左侧的独立的SRAM中缓冲)，prgramdac(用来对存有8Bit色彩表的Ramdac编程)，vgacore(创建显示器VGA同步信号或其它控制信号)。

项目的主体部分是控制一个视频解码器(Video Decoder)，其型号是SAA7113，就是用它来完成将接收到的NTSC，SECAM，或者PAL制式的视频信号数字化。数字化后的视频信号通过VPO总线送到FPGA。这种数字化的视频数据的传送是用由视频解码器(Video Decoder)发出的行锁时钟LLC(Line-locked Clock)的上升沿同步的。而FPGA使用**I2C**总线（即串行数据线（SDA）和串行时钟线（SCL））对SAA7113的视频选项编程。

¿2.2 关于I2C

I2C 总线在 XSV800 Board 上被普遍使用，因此为完成该项目我必须对该总线结构有一个全面地了解。

串行扩展总线在单片机系统中的应用是目前单片机技术发展的一种趋势。在目前比较流行的几种串行扩展总线中，I2C 总线以其严格的规范和众多带 I2C 接口的外围器件而获得广泛的应用。I2C 总线是由 PHILIPS 公司推出的芯片间串行传输总线。它以 1 根串行数据线（SDA）和 1 根串行时钟线（SCL）实现了全双工的同步数据传输。随着 I2C 总线研究的深入，它已经广泛应用于视/音频领域、IC 卡行业和一些家电产品中，在智能仪器、仪表和工业测控领域也越来越多地得到应用。I2C 总线的广泛应用是同它卓越的性能和简便的操作方法分不开的。

（1） 硬件结构上具有相同的硬件接口界面。I2C总线系统

中，任何一个I2C总线接口的外围器件，不论其功能差别有多大，都是通过串行数据线（SDA）和串行时钟线（SCL）连接到I2C总线上。这一特点给用户在设计应用系统中带来了极大的便利性。用户不必理解每个I2C总线接口器件的功能如何，只要将器件的SDA和SCL引脚连到I2C总线上，然后对该器件模块进行独立的电路设计，从而简化了系统设计的复杂性，提高了系统抗干扰的能力，符合EMC (Electromagnetic Compatibility)设计原则。

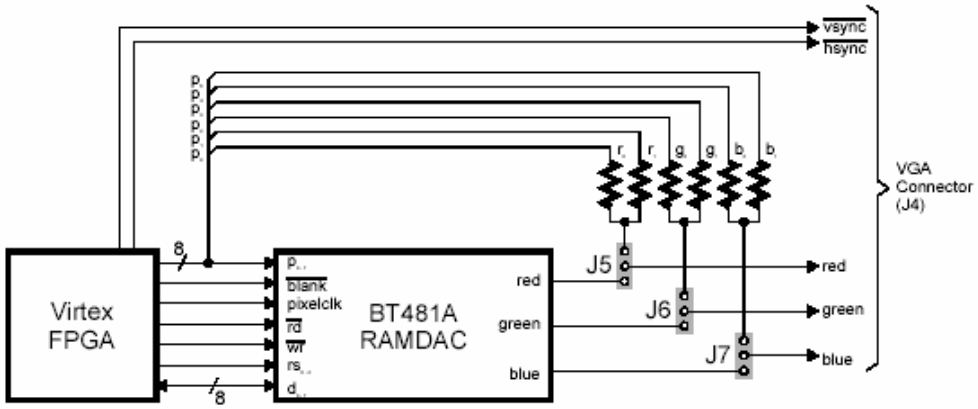
（2） 总线接口器件地址具有很大的独立性。在单主系统中，每个I2C接口芯片具有惟一的器件地址，由于不能发出串行时钟信号而只能作为从器件使用。各器件之间互不干扰，相互之间不能进行通信，各个器件可以单独供电。MCU与I2C器件之间的通信是通过独一无二的器件地址来实现的。

（3） 软件操作的一致性。由于任何器件通过I2C总线与MCU进行数据传送的方式是基本一样的，这就决定了I2C总线软件编写的一致性。

（4） PHILIPS公司在推出I2C总线的同时，也为I2C总线制订了严格的规范，如：接口的电气特性、信号时序、信号传输的定义等。规范的严密性，结构的独立性和硬、软件接口界面的一致性，极大地方便了I2C总线的设计。模块化和规范化，伴随而来的是用户在使用I2C总线时的“傻瓜”化。

¿2.3 PrgRamdac程序及其控制的RAMDAC

RAMDAC与FPGA的连接如下图**(图2.1)**所示，用这个连接方式来向外提供一个VGA接口。FPGA产生的视频信号可以直接显示或通过一个BT481A RAMDAC在一台VGA显示器上显示。



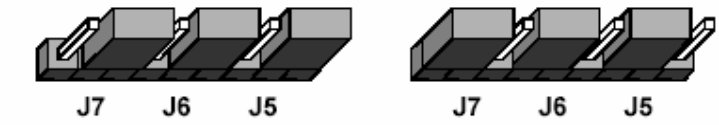
(图2.1)

FPGA直接产生VGA信号时，PBUS的低六位向一个单独的电阻梯度的DAC提供红绿蓝各两位的色彩信息。DAC输出与FPGA输出的水平/垂直同步信号一起送入显示器中。

主要讨论通过RAMDAC的情况。RAMDAC产生VGA色彩信号，然后FPGA用全部的8位P-BUS来传送当前像素的色彩索引信号。而色彩索引是用以查询储存在RAMDAC芯片中的色彩表中的24位的色彩值(红，绿，蓝各八位)。这种通过P-BUS的传输是用FPGA产生的像素时钟(PIXELCLK)来同步的。当像素落到显示器的希望的显示区域以外时，FPGA就降低/Blank信号使之有效。

FPGA使用D-BUS连同/WR，/RS，/RD信号来初始化RAMDAC的色彩表，24位色彩表单元在三选(RS0,RS1,RS2)的分段传输记录中传递用于些色彩表。在最后一位色彩信息到达D-BUS后，分段传输记录的内容被写入色彩表，然后内嵌色彩表地址变量指向下一个单元。

使FPGA直接产生或通过RAMDAC产生VGA信号的跳线方式如下图**(图 2.2)**所示：



直接VGA输出跳线　　RAMDAC输出跳线

(图 2.2)

FPGA与VGA信号产生电路连接的管脚分配如下图**(图 2.3)**所示。RAMDAC和以太网接口芯片共享其中一些连接。仅在系统初始化后RAMDAC管脚用于读取色彩表参数。另外一些管脚被用于以太网数据传输和接收，通常仅在系统初始化后激活。

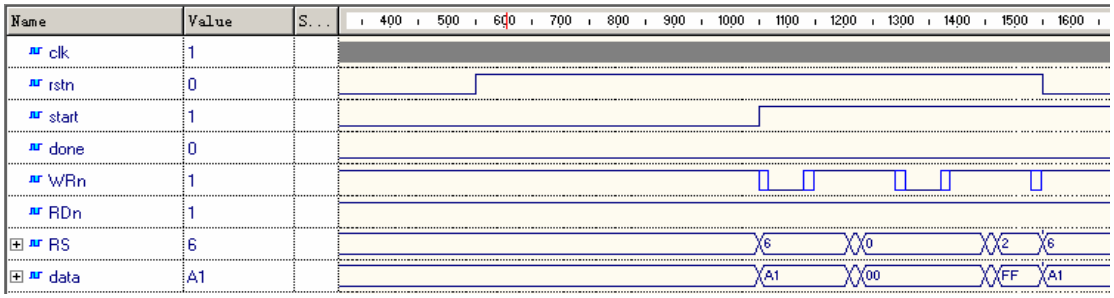
Direct VGA Pin	RAMDAC Pin	Virtex FPGA Pin	LXT970A Function
	PIXELCLK	52	
/HSYNC	/HSYNC	48	
/VSYNC	/VSYNC	49	
	/BLANK	50	
RED0	P0	70	
RED1	P1	71	
GREEN0	P2	72	
GREEN1	P3	73	
BLUE0	P4	74	
BLUE1	P5	78	
	P6	79	
	P7	80	
	/RD	47	
	/WR	46	
	RS0	31	TXD4
	RS1	28	RX_ER
	RS2	26	RX_DV
	D0	42	TXD0

(图 2.3)

以下是PRGRAMDAC程序的VHDL源代码，用以对FPGA编程控制RAMDAC工作的，模块Port定义部分如下：

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity prgramdacver2 is
port (
clk: in STD_LOGIC;                                  -- 时钟
rstn: in STD_LOGIC;                                 -- 低有效异步重置
start: in STD_LOGIC;                                -- 开始信号
done: out STD_LOGIC;                                -- 表明程序完成
WRn: out STD_LOGIC;                                 -- RAMDAC写选通线(低有效)
RDn: out STD_LOGIC;                                 -- RAMDAC读选通线(低有效)
RS: inout STD_LOGIC_VECTOR (2 downto 0);    -- RAMDAC记录选通线
data: inout STD_LOGIC_VECTOR (7 downto 0)   -- RAMDAC双向数据线
        );
    end prgramdacver2;
```

关于功能描叙的代码相当长而且十分的复杂，故这里未提供这些源代码。以下是对这个模块的行为级仿真的波形图**(图2.4)**：



(图2.4)

¿2.4 VGACORE程序解析

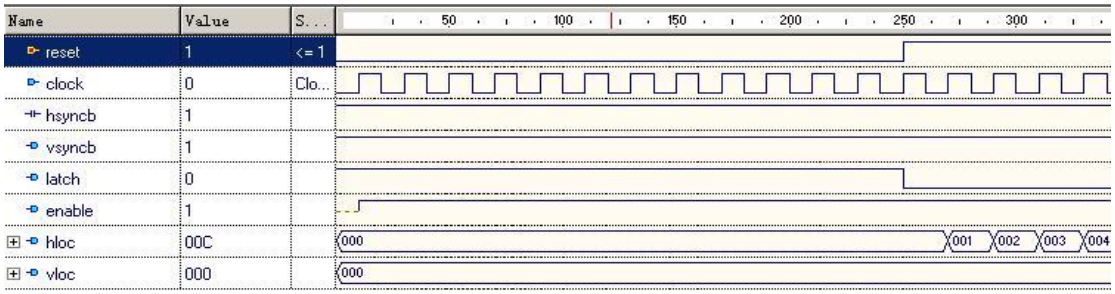
这个程序是FPGA用来产生VGA显示的同步信号及其它控制信号。这些对于显示的控制，也就是分辨率和刷新率。缺省的显示设置是分辨率：800*600，刷新率：72Hz。而通过这个程序可以很方便的改变这些设置。改变它们也就意味着改变了系统时钟频率，而缺省频率是50MHz。

这一程序的Port定义如下(功能描叙部分的源代码省去未给出)：

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
use IEEE.std_logic_unsigned.all;
entity vgacore is
            generic (
H_SIZE : integer := 800;                -- 输入图像的水平像素，最大800
V_SIZE : integer := 600                 -- 输入图像的垂直像素，最大600
          );
                port (
reset: in std_logic;                     -- 异步重置，低有效
clock: in std_logic;                     -- 时钟
hsyncb: buffer std_logic;                -- 水平线同步
vsyncb: out std_logic;                   -- 垂直帧同步
latch: out STD_LOGIC;                    -- 锁定新的三原色值
enable: out STD_LOGIC;                   -- 三原色输出线使能/接地
hloc: out std_logic_vector(9 downto 0); -- video RAM的水平地址编码
vloc: out std_logic_vector(9 downto 0)  -- video RAM的垂直地址编码
);
      end vgacore;
```

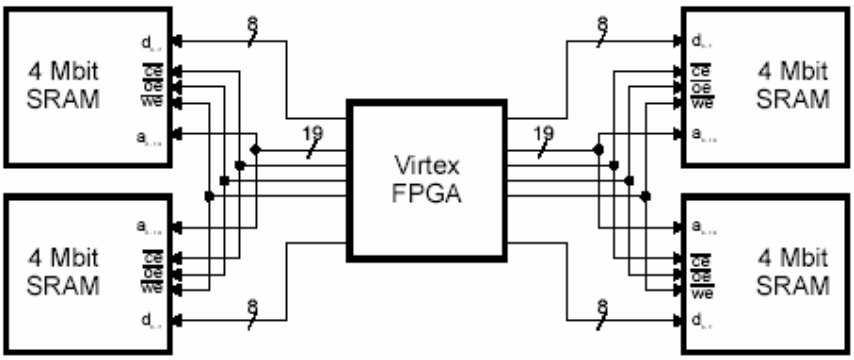
以下是该模块的行为级仿真波形图**(图 2.5)**：



(图 2.5)

¿2.5 sram512kleft16bit50mhz-sv01b程序解析

FPGA可以使用如下图**(图 2.6)**所示的两组独立的**SRAM**：每组**SRAM**都是由**512K*16Bits**构成，这是两块**Winbond**的**AS7C4096**的**512K*8Bits**的**SRAMs**。与**SRAM**连接的**FPGA**的管脚连接在电路图后的附图**(图 2.7)**提供：



(图 2.6)

SRAM Pin	Virtex FPGA Pin to Left Bank	Virtex FPGA Pin to Right Bank
/CE	186	109
/OE	228	95
/WE	201	68
D0	202	70
D1	203	71
D2	205	72
D3	206	73
D4	207	74
D5	208	78
D6	209	79
D7	215	80
D8	216	81
D9	217	82
D10	218	84
D11	220	85
D12	221	86
D13	222	87
D14	223	93
D15	224	94
A0	200	67
A1	199	66
A2	195	65
A3	194	64
A4	193	63
A5	192	57
A6	191	56
A7	189	55
A8	188	54
A9	187	53
A10	238	108
A11	237	107
A12	236	103
A13	235	102
A14	234	101
A15	232	100
A16	231	99
A17	230	97
A18	229	96

(图 2.7)

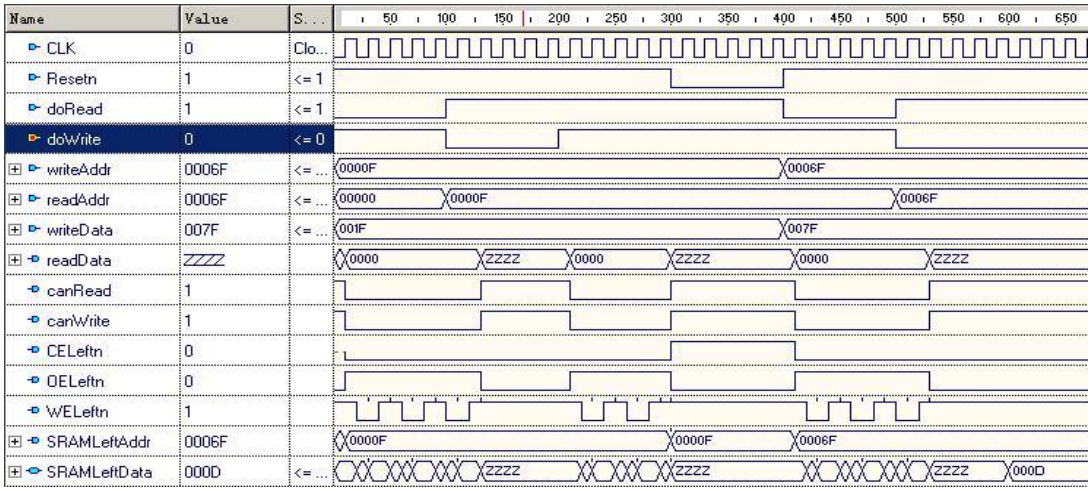
这种连接在XSV800 Board上提供一个单独的面向SRAM的界面，这个程序提供了一个RAM控制器。用以控制数字化的视频数据向左侧的RAM中存取，也既是为这些数据提供一个缓冲区。

这个模块的Port定义部分的代码如下所示：

```
library IEEE;
use IEEE.std_logic_1164.all;

entity sraminterface is
        port (
CLK: in STD_LOGIC;                                  -- 时钟
Resetn: in STD_LOGIC;                               -- 异步重置(低有效)
doRead: in STD_LOGIC;                               -- 当前无用
doWrite: in STD_LOGIC;                              -- 执行写操作
readAddr: in STD_LOGIC_VECTOR (18 downto 0);    -- 读地址
writeAddr: in STD_LOGIC_VECTOR (18 downto 0);   -- 写地址
readData: out STD_LOGIC_VECTOR (15 downto 0);   -- 读数据
writeData: in STD_LOGIC_VECTOR (15 downto 0);    -- 写数据
canRead: out STD_LOGIC;                             -- 执行读操作(为1时读)
canWrite: out STD_LOGIC;                             -- 执行写操作(为1时写)
CELeftn: out STD_LOGIC;                             -- 左侧的SRAM的CEn信号
OELeftn: out STD_LOGIC;                             -- 左侧的SRAM的CEn信号
WELeftn: out STD_LOGIC;                             -- 左侧的SRAM的WEn信号
SRAMLeftAddr: out STD_LOGIC_VECTOR (18 downto 0);   -- 左侧SRAM的地址总线
SRAMLeftData: inout STD_LOGIC_VECTOR (15 downto 0);   -- 左侧SRAM的数据总线
    );
    end sraminterface;
```

以下是这个模块的行为级仿真波形图(图2.8)：

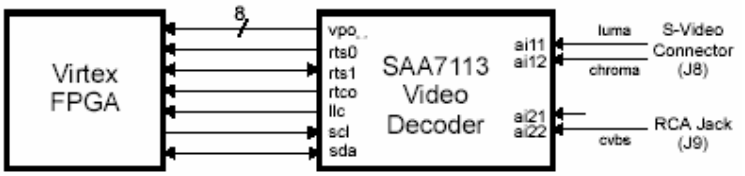


(图2.8)

¿2.6 saa7113程序

这个程序是项目VideoIn的最顶层(Top-Level)的程序，其控制着视频信号输入到saa7113芯片后解码后送入FPGA全过程。FPGA通过使用VPO总线来将数据编码读入。而FPGA从时间控制代码中获取读数据的时间和些数据的时间。

以下是saa7113 Video Decoder与FPGA的连接图**(图2.9)**：



(图2.9)

解码器与FPGA的管脚连接如下图**(图2.10)**：

SAA7113 Pin	Virtex FPGA Pin
LLC	92
RTS0	111
RTS1	110
RTCO	113
VPO0	116
VPO1	117
VPO2	118
VPO3	125
VPO4	126
VPO5	127
VPO6	128
VPO7	130
SCL	114
SDA	115

(图2.10)

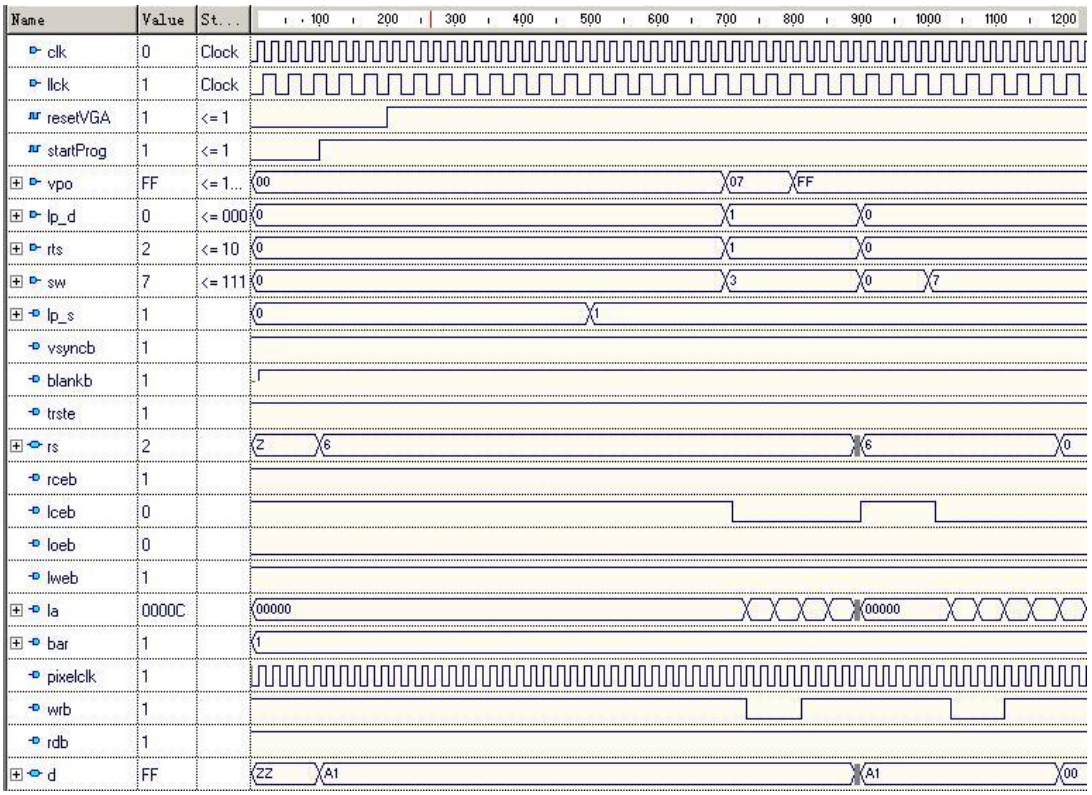
该设计中包含了以上提及的所有的三个子模块：prgramdac，vgacore，sram512kleft16bit50mhz-sv01b。

这个顶层模块的Port定义部分的代码如下：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity saa7113 is
    port(
clk: in std_logic;                          -- 50 MHz时钟
lp_d: in std_logic_vector(2 downto 0);      -- 并口输入
lp_s: out std_logic_vector(3 downto 3);     -- 并口输出
sw: in std_logic_vector(3 downto 1);        -- 按钮开关
scl: inout std_logic;                       -- 双向I2C时钟信号
sda: inout std_logic;                       -- 双向I2C数据信号
llck: in std_logic;                         -- SAA7113视频时钟(27 MHz)
vpo: in std_logic_vector(7 downto 0);       -- SAA7113数据信号
rts: in std_logic_vector(1 downto 0);       -- 实时视频信号
hsyncb: buffer std_logic;                   -- 水平同步
vsyncb: out std_logic;                      -- 垂直同步
blankb: out std_logic;                      -- 视频阻断
trste: out std_logic;                       -- 网络三态线
bar: out std_logic_vector(7 to 9);          -- 状态显示
p: out std_logic_vector(7 downto 0);        -- RAMDAC像素总线
pixelclk: out std_logic;                    -- 像素时钟
wrb, rdb: out std_logic;                    -- 读/写闸门 to BT481 RAMDAC
d: inout std_logic_vector(7 downto 0);      -- 记录RAMDAC数据总线
rs: inout std_logic_vector(2 downto 0);     -- 记录RAMDAC地址总线
rceb, lceb, loeb, lweb: out std_logic;      -- SRAM控制线
la: out std_logic_vector(18 downto 0);      -- SRAM地址总线
ld: inout std_logic_vector(15 downto 0) -- SRAM数据总线
    );
    end saa7113;
```

这段代码中的BUFG模块是一个输出缓冲门，IOBUF_F_12这个模块是一个带控制端T，还带一个输入输出双向线IO，以下为VideoIn项目总的行为级仿真波形图**(图2.11)**：



（图 2.11）

第三章 通过 VGA 接口输出视频

本章将介绍如何向 FPGA 编程来控制系统将数字化后的 NTSC，SECAM 或者 PAL 制式的视频信号，按 VGA 制式输出信号。

¿3.1 概述

在 Active HDL 上创建一个新的项目，实验中我将其命名为 VGAOut，用以对 FPGA 编程，使它能够将捕获到的视频信号通过 VGA 接口输出。它使用 XSV800 Board 上带着的 RAMDAC 设计一个面向 VGA 显示器的独立的 VGA 接口。这个项目还包含着对一个可编程的 256 色的色彩表和 15 位高彩显示的支持。而程序的缺省的配置方案是：50MHz 时钟控制，分辨率 800*600，刷新率 72Hz，而且应该能够通过该程序很容易的改变这些配置方案。

这个项目的最上层是一个叫 VGA 的模块，还包含着两个子模块：VGACORE 和 PRGRAMDAC。

¿3.2 关于 VGACORE 和 PRGRAMDAC 程序

VGACORE 和 PRGRAMDAC 这两个模块的程序和上述的 VideoIn 项目中的两个同名的模块是一样的，这里就不在说明了。它们在这里的作用如下：

VGACORE：

这里的 VGACORE 程序在这个项目中主要是为 VGA 显示提供垂直同步线(vsync)和水平同步线(hsync)。垂直同步线(vsync)将水平同步线(hsync)溢出定时关闭，从而垂直同步线(vsync)成为简化的时序逻辑。这些时序可以影响显示的分辨率和刷新率，而且通过该程序应该可以很方便的修改这些参数。

注意：不正确的设置这些参数会使得图像变形或是根本没有图像显示。标准时序提供了在每一种分辨率和刷新率下所需要的水平时序(以给定的时钟或它分频后的时钟信号)和垂直时序(在水平时序溢出时关闭)。

PRGRAMDAC：

虚拟色彩(Pseudo Colour)模式：这个程序的一个或者说是 RAMDAC 的一个选项是通过一个可编程的 256 色的色彩对照表

来创建颜色。在创建颜色前，必须通过 3Byte 的色彩值来对色彩表编程。创建一个特定的颜色时，色彩表上的本地数据将通过像素总线送入 RAMDAC。程序将解析色彩表模式和高彩色模式之间选择转换时所需要的线。

高彩色(High Color)模式： 通过向RAMDAC发送一个2Byte的数据(15位数据可用：每5Bits决定一种基本色，加一位校验位)调用可能产生的32’768种色彩。在像素线上发送2Byte数据意味着向RAMDAC传送数据的速率必须是两倍于时钟频率，换句话说在行锁时钟LLC(Line-locked Clock)为27MHz的时候，发送数据的时钟应该是50MHz。与上述虚拟色彩模式相比，这个替代方案将使用单边沿的输入模式，将数据分开发送。

操作时，每种不同的VGA模式的顶层文件都将生成一个可旋转的测试模式。以下就是输入的映射开关：

虚拟色彩(Pseudo Colour)：

DIP 1: 控制测试模式是否旋转
DIP 2: 控制旋转的方向
DIP 3: 控制测试模式是否水平或垂直

高彩色(High Colour)：

DIP 1-2: 控制测试模式中的红色部分的组成。
DIP 3-4: 控制测试模式中的绿色部分的组成。
DIP 5-6: 控制测试模式中的蓝色部分的组成。
DIP 7: 控制测试模式是否进行水平旋转。
DIP 8: 控制测试模式是否进行垂直旋转。
PB 1: 控制水平旋转方向。
PB 2: 控制垂直旋转方向。

¿3.2 VGA 程序（虚拟色彩(Pseudo Colour)模式）

这是 VGAOUT 项目的最主要部分，使这个项目的最顶层设计(Top-Level Design)。它负责统一协调 VGA 信号输出这项工作的全过程，这个模式采用的是虚拟色彩模式(Pseudo Colour)来调用颜色，也就是 256 色彩表的选色模式。

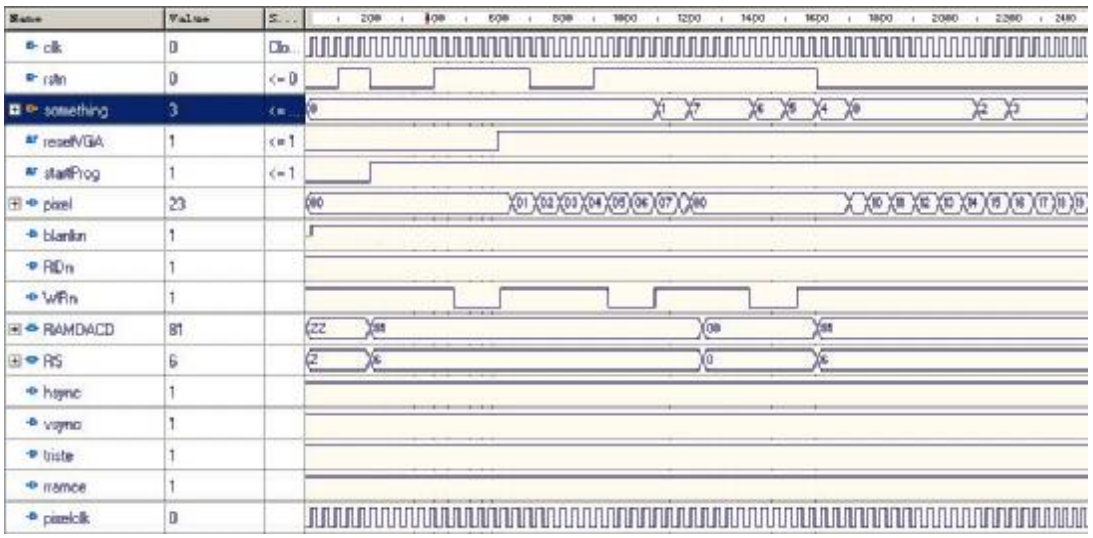
这段程序的 Port 定义说明部分如下：

```
ibrary IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity vga is
    port (
```

```
clk: in STD_LOGIC;                                    -- 时钟
rstn: in STD_LOGIC;                                   -- 异步重置，低有效
something: in STD_LOGIC_VECTOR(2 downto 0);           -- 各种方式影响测试模式
pixel: out STD_LOGIC_VECTOR (7 downto 0);             -- RAMDAC 像素线
blankn: out STD_LOGIC;                                -- RAMDAC blank 信号
RDn: out STD_LOGIC;                                   -- 连接 RAMDAC RDn
WRn: out STD_LOGIC;                                   --连接 RAMDAC WRn
RAMDACD: inout STD_LOGIC_VECTOR (7 downto 0);         -- RAMDAC 数据线
RS: inout STD_LOGIC_VECTOR (2 downto 0);              -- RAMDAC RS 线
hsync: out STD_LOGIC;                                 -- 接显示器水平同步
vsync: out STD_LOGIC;                                 -- 接显示器垂直同步
triste: out STD_LOGIC;                                -- 三态的以太网 PHY 信号
rramce: out STD_LOGIC;                                -- 右侧 RAM 芯片使能
pixelclk: out STD_LOGIC                               -- RAMDAC 像素时钟
    );
end vga;
```

具体的处理程序很长，这里省去了，不在一以解释。对这种模式我在 Active HDL 上对它进行了行为级的仿真，得到的波形图如下图**(图 3.1)**：



(图 3.1)

¿3.3 HICOLVGA 程序（高彩色(High Color)模式）

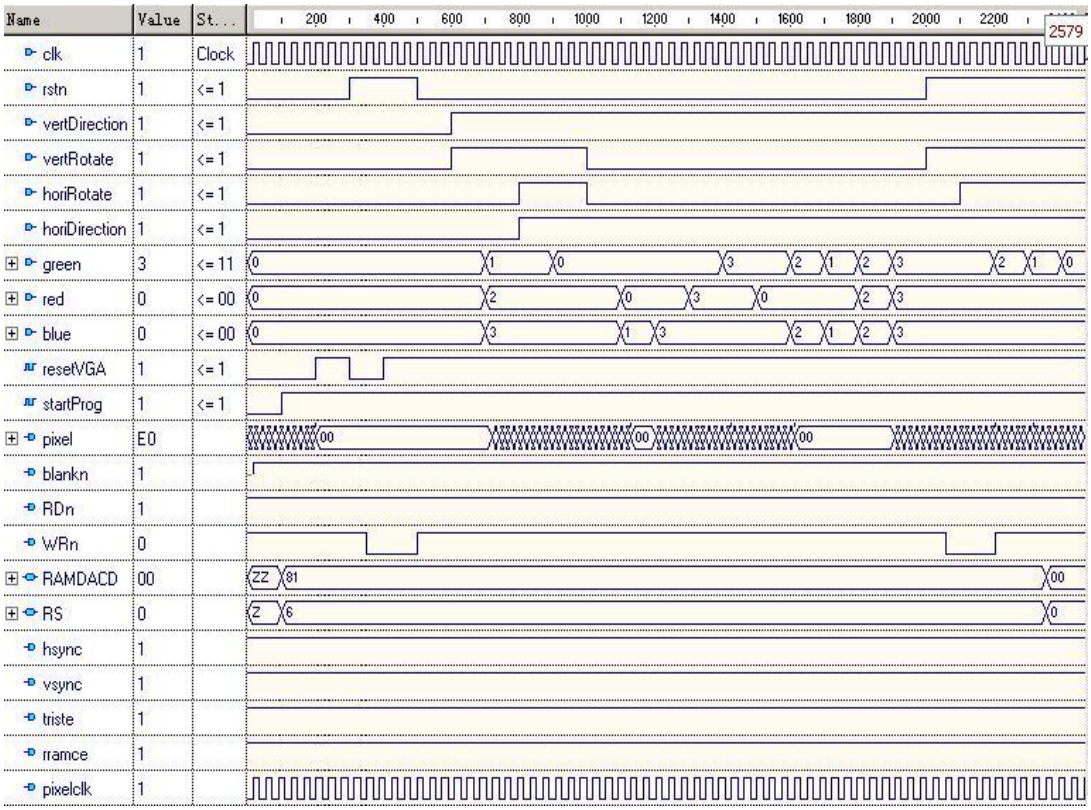
这个程序的作用和上一节中的 VGA 程序是相同的，只是它采用的是用高色彩模式(High Colour)来调用颜色。

Port 定义说明部分如下：

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity vga is
      port (
clk: in STD_LOGIC;                              -- 时钟
rstn: in STD_LOGIC;                             -- 异步重置，低有效
red: in STD_LOGIC_VECTOR(1 downto 0);   --控制红色部分的组成的开关。
green: in STD_LOGIC_VECTOR(1 downto 0);  --控制绿色部分的组成的开关。
blue: in STD_LOGIC_VECTOR(1 downto 0);  --控制蓝色部分的组成的开关。
vertRotate: in STD_LOGIC;                       -- 为 1 时，测试模式垂直旋转。
horiRotate: in STD_LOGIC;                       --为 1 时，测试模式水平旋转
vertDirection: in STD_LOGIC;                    -- 垂直转换方向
horiDirection: in STD_LOGIC;                    -- 水平转换方向
pixel: out STD_LOGIC_VECTOR (7 downto 0);     -- RAMDAC 像素线
blankn: out STD_LOGIC;                          -- RAMDAC blank 信号 l
RDn: out STD_LOGIC;                             --连接 RAMDAC RDn
WRn: out STD_LOGIC;                             --连接 RAMDAC WRn
RAMDACD: inout STD_LOGIC_VECTOR (7 downto 0);   -- RAMDAC 数据线
RS: inout STD_LOGIC_VECTOR (2 downto 0);          -- RAMDAC RS 线
hsync: out STD_LOGIC;                           -- 接显示器水平同步
vsync: out STD_LOGIC;                           -- 接显示器垂直同步
triste: out STD_LOGIC;                          --三态的以太网 PHY 信号
rramce: out STD_LOGIC;                          --右侧 RAM 芯片使能 pixelclk:
out STD_LOGIC                                   -- RAMDAC 像素时钟
      );
end vga;
```

以下是行为级仿真波形图**(图 3.2)**：



(图 3.2)

¿3.4 小结

对这个项目有几点扩展和修正。在显示器上创建的显示不清晰——水平线有轻微的闪烁，这种现象通过更改分辨率，刷新率是不能消除掉的，所以引起这种现象的原因显然不是上述的这些因素，也许带参量限制的显示方式可能减少或除去这种现象。另外，还有一些修正方案可以增加该程序对各种不同的分辨率和刷新率的支持。

对这个设计来说还有些事情是必须特别注意的：高彩模式的显示必须以高消耗来实现，否则在双边沿时钟控制的显示模式中，会有短时脉冲波形干扰(低频干扰)——表现为出现假脉冲。PHY 设备的一些输出和 RAMDAC 的编程线是共用的，所以输出肯定会由于 PHY 三态输入的低而被 Disable 掉的。

第四章 测试

这章讲叙的是如何测试这块 XSV800 Board 的功能。也就是要验证它的逻辑功能是否能准确地实现，看看它能否将输入的视频信号(NTSC，SECAM 或者 PAL 制式)转换成 VGA 信号输出。

¿4.1 使用VideoIn项目

将以上的程序生成的二进制(.Bit)文件下载到这块测试板上(可以通过串/并口Parallel/Serial Port或者Xchecker电缆)，而项目VideoIn文件包中带着的批处理文件Saapal.bat必须在与测试板相连的计算上运行，用以对视频解码器芯片(video decoder chip)saa7113编程。在这些完成以后按下开关3开始捕获帧，按下开关2就停止捕获。停止时被捕获的最后一帧图像会留于显示。缺省模式允许高彩色的图像尽可能快的刷新。这一捕获过程会在显示器引起很多杂音。代码的一些内容被注释掉了，如果是采用以下的特征来实现捕获就可以取消这些注释使之成为语句：

降低捕获率：降低捕获率后，每幅图像(帧)在刷新的时候可以看得更加清晰。

显示器抑止全部或部分显示图象的显示：在新的一帧被捕获时，显示器通常不会抑止图象的显示。而在停止向RAM中读取数据而向RAM中写显示数据的时候，帧捕获会引起信号噪声。这种情况有可能在帧捕获时抑止全部或部分图象的显示，这将导致图像重新显示时之间黑屏。

变成8位数据：在改成使用8位数据后，就可以在显示屏上描绘更多的像素。默认的显示宽度是720像素，每串数据映射两个毗邻的像素。这个模式就如同是360“像素”(两倍于常规像素宽度)。使用8位像素，RAM中的每个Byte存储着两个像素，全部的720个像素宽度都能被很方便的调用。

试验表明：精确的时钟频率是没有必要的，通过程序设置不同于默认配置的用户配置方案(分辨率和刷新率)在经过反复试验后证实是可行的。

¿4.2 用saa7113程序对解码器编程

这是一个由XESS公司提供的可执行程序，用来对saa7113

解码器芯片配置。其使用命令行参数。

用下面这种格式向SAA7113芯片的一个子地址中写入数据：

saa7113 w sub-address value

所有的sub-address value都是采用16进制数表示。

用如下的这种格式从SAA7113芯片的一个子地址中读取数据

saa7113 r sub-address value

地址值以及从地址中读取到的数据是16进制的整型。这个从解码器芯片saa7113中返还的值将被用于显示。

¿4.3 测试中的问题和解决方案

前面已经提到过，在这个设备捕获帧时，由于RAM的的争用，会引起信号短时脉冲波形干扰(低频干扰)。这点可以藉由降低VGA输出时序的像素时钟到25MHz和智能隔行扫描以及RAM读写的流水线操作以产生一个输入到输出的即时满流信号就可以解决这个问题。隔行扫描也会引起一些显示问题：因为芯片输出的两个场取样不同步时，会引起屏幕捕获的一半水平线被另外一半取代了。

这个项目使用的RAM控制器也将会引发一些问题：有一些写入会被忽略掉。这将在一些帧捕获时出现错误。

对这个项目做一些扩展，就可以明显的改进这个项目功能的使用。这个项目的代码写得不够清晰，而且还有很多问题需要修正。

总结

本次论文的主要目的是了解 XSV800 Board 的视频部分的使用，并验证其视频部分的逻辑功能。我从这块测试板上的可用资源开始介绍，了解我们在使用它时可以直接调用的部分。然后开始介绍将 NTSC，SECAM 或者 PAL 制式的视频信号解码的项目，这里主要会使用 XSV800 Board 上一个视频解码器 SAA7113。由于 FPGA 对视频解码器编程使用的是 I2C 的总线结构，而这种总线结构也是一种使用相当广泛的结构，因此我重点介绍了一下这种总线。接着我由下向上介绍了输入视频信号处理部分的程序源代码(VHDL)，这些对 XSV800 Board 上 FPGA 的编程使其能够控制系统完成对 NTSC，SECAM 或者 PAL 制式的视频信号解码。我对这个项目的各个子模块以及整个项目都在 Active HDL 上作了行为级的仿真，主要是为了验证这些模块的功能是否正确，整体的功能是否能实现。本文也给出了这些仿真波形图。然后介绍了一下如何将前面前一步中数字化的视频信号重新编码，然后通过 VGA 接口向显示器输出。这里我同样在 Active HDL 上对各模块做了行为级的仿真，以验证其功能。最后介绍了一下将上述程序生成的二进制文件下载到 XSV800 Board 上对 FPGA 编程后，对其进行实际验证的过程，方法以及遇到的问题。

感谢

首先我要感谢大学四年中所有教过我的老师。他们勤勤恳恳的工作使我受益匪浅，从他们身上我不仅学到了知识，更领悟了对待工作的态度。

然后我要感谢唐长文老师，作为指导老师，在我完成论文的过程中对我提供的热心的辅导和帮助。很多疑难问题，我都在他那里都得到了快速的解答。

另外，我要向我同寝室的同学表示歉意，我的通宵工作可能给他们的生活带来了不便，影响了他们的休息。在这里我再次表示感谢和深深的歉意。

再次感谢以上的老师和同学。

参考书目

- **Skahill，“VHDL for programmable logic”**
- **复旦大学电子工程系出版，“可编程逻辑器件设计”**
- **刘晓伟，“VGA、SVGA 彩色显示的原理、维修及图集”**
- **Sezan，“Image and video processing”**
- **何立民，“I2C 总线应用系统设计”**