# 华中科技大学

# 《电子线路设计、测试与实验》实验报告

实验名称：　　基于 FPGA 的图像处理系统

院（系）：　　电子信息与通信学院

专业班级：　　电信提高 1x01

姓名：　　伍 xx　Xxx　Xxx

学号：　　U201xxxxxx

　　U201 xxxxxx

　　U201 xxxxxx

时间：　　2017.6.7

地点：　　XXXXXX

实验成绩：

指导教师：　　XXX 教授

2017 年　6 月 9 日

# 目录

# 一 实验目的

1.熟练掌握基础的硬件开发技能，实现一个完整的有一定应用价值的工程项目
2.基于 Verilog 语言和 FPGA 的资源，实现对图像信号的显示和处理

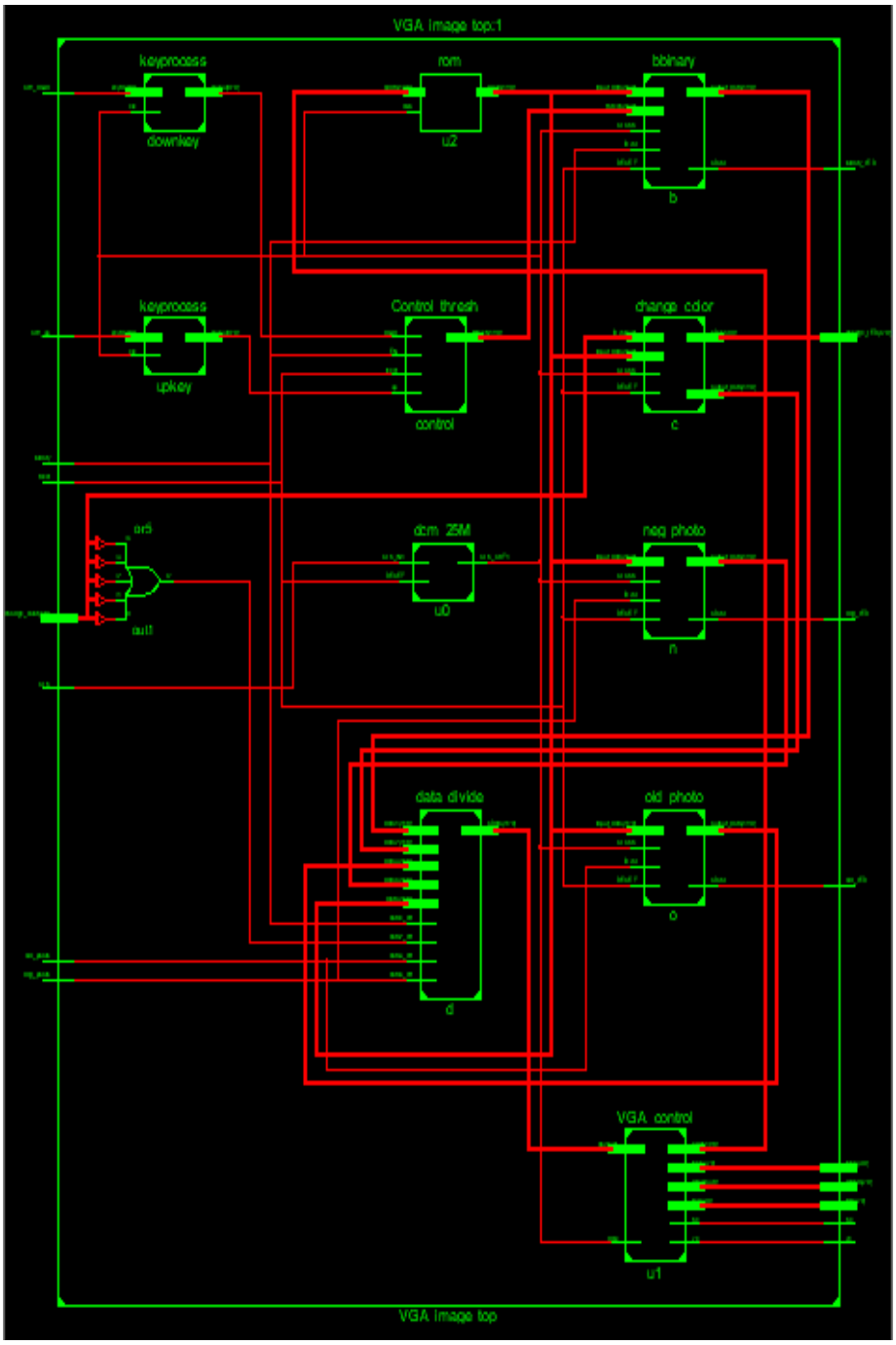
# 二 运行环境

实验全部采用 Verilog 语言进行编程调试，工程文件的编译环境为 ISE 14.7，FPGA 开发板使用 Xlinx 提供的 Nexys 4 DDR 开发板

# 三 实验原理与电路设计

## 3.1 顶层结构设计

系统可以分为时钟管理、图像存储、图像处理、VGA 显示、数据选择等几个模块组成。时钟管理模块使用 ISE 14.7 IP 核配置锁相环对 Nexys 4 板载的 100M 时钟分频，获得适合 VGA 显示的时钟频率；图像储存模块同样用 IP 核配置，采用单口 ROM 一次性写入需要处理的图像文件；VGA 显示模块对 VGA 时序进行控制，读取 ROM 内信息并输出至显示屏；图像处理模块又按功能分为多个模块，对 ROM 输出的图像数据进行处理后再传输至 VGA 显示模块；最后，由于有多种处理功能，可能的处理结果也是多种的，因此在图像处理模块与 VGA 显示模块间还需要一个起数据选择器作用的数据选择模块。最终设计得到的电路结构如下图所示。




## 3.2 IP 核配置

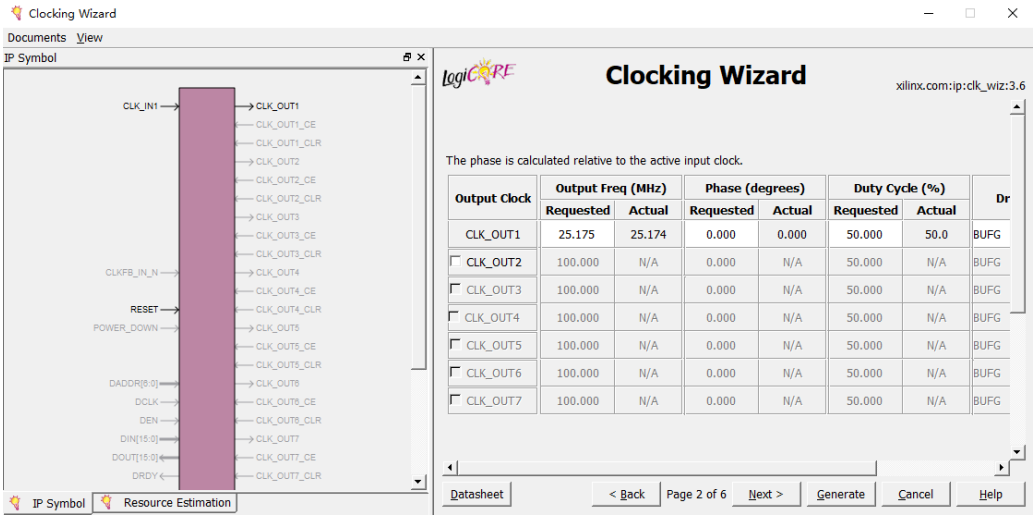
### 3.2.1 锁相环配置

在工程文件中选择 New Source，新建 IP core 文件，在 FPGA Features and Design 下找到 Clocking，进入锁相环配置向导。

进入向导界面后，设置 Input Clock 为 100MHz，OUT Clock 为 25.175MHz，此处输出频率是根据之后的 VGA 输出分辨率和刷新频率计算得到的。

选择可选的输入输出端口如 RESET 和 LOCKED，这里我们选择只需要 RESET 端。其余设置均保持默认。

点击 generate 即可配置得到我们需要的时钟分频模块。

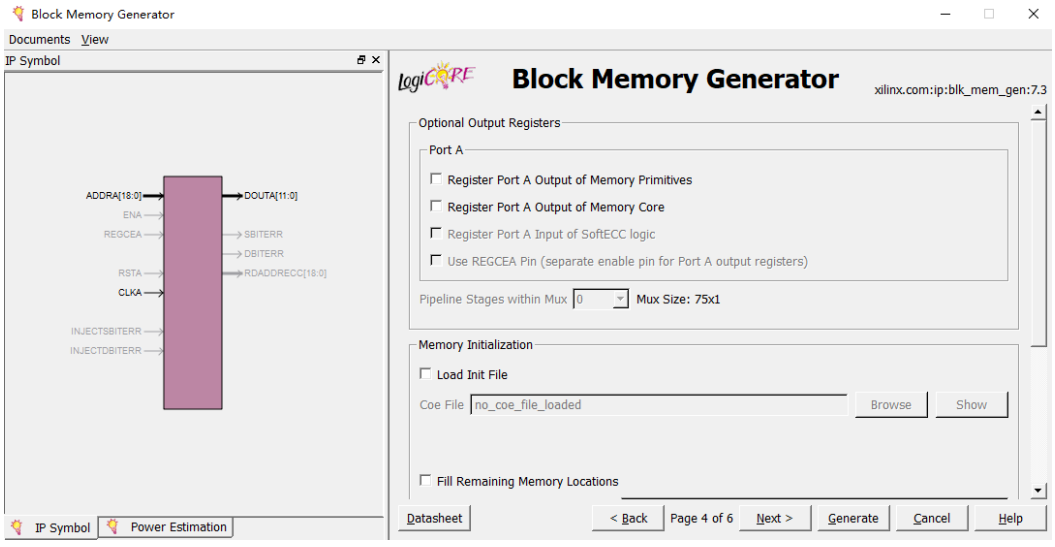



### 3.2.2 ROM 配置

由于 Nexys 4 自带的外部储存器 SDRAM 调用不方便，而芯片 Artix 7 配置得到的 ROM 足以存放我们此次实验所需的单张图片。因此我们同样用新建 IP core 文件的方式，在 Memories & Storage Elements 下找到 RAMs & ROMs，进入 ROM 配置向导。

首先，选择储存器类型为 Single Port ROM。然后设置 Read Width 为 12，对应 Nexys 4 的 444VGA 接口，设置 Read Depth 为 307200( = 640 x 480)，对应所需图片的像素点个数。

最后 Load Init File，将所需显示的图片文件处理成*.coe 文件，导入到配置向导中。将图片文件处理成*.coe 文件可以采用 Matlab 实现，具体代码见附录。其余设置同样保持默认。若文件大小合适且无出错，则可以 generate 得到存有所需图像信息的 ROM 储存器。这一步可能耗费很长时间，需要耐心等待。

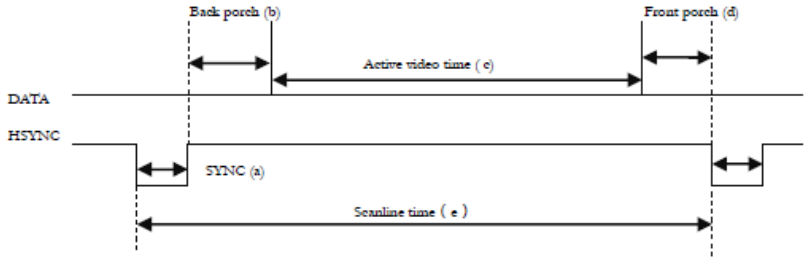



## 3.3 VGA 图像显示

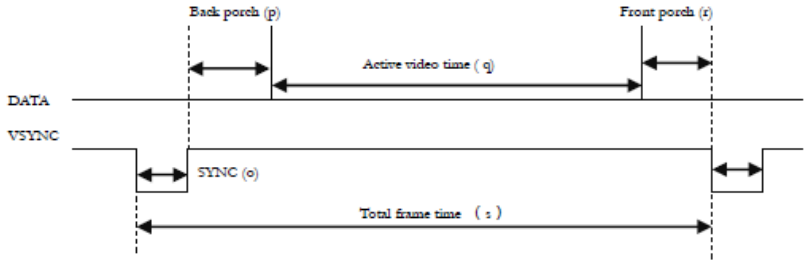
VGA 接口是一种 15 针引脚的视频传输接口，Nexys 4 提供了对 R，G，B，HS，VS 共 5 个引脚的 IO 口，其中，RGB 三色输出为模拟信号输出，在编程时，Nexys 4 内置了 D-A 转

换，将 RGB 各四位的数字信号转化会模拟信号输出。因此，通过编写适当的 HS 和 VS 时序并输出 RGB[11:0]的图像颜色信号，即可在 VGA 显示屏上显示图像。下面以 640x480@60Hz 为例讲解如何使用 VGA 显示图片。

HS 是行扫描信号，VS 为帧扫描信号。两者的时序四个时期，单纯只为显示图像不需要对四个时期有过多的了解。简单来讲，HS 信号在高电平时，有一段有效的时期，时长为 640 个时钟周期。即 HS 信号每周期对应扫描一行像素点。类似的，VS 信号每周期对应一帧像素点，一个周期的高电平时期有一段时长为 480 个 HS 周期的有效时间。在有效时间内，每过一个时钟周期，VGA 对 ROM 的读取地址就加一，RGB[11:0]从数据选择器更新一次输出，一帧扫描结束，地址重新置零。如此即可实现对 VGA 的显示。HS 和 VS 的时序示意图和对于不同的分辨率，HS 和 VS 的时序参数如下两图。




VGA 行数据时序




VGA 帧数据时序

常见刷新率时序表：

| 显示模式 | 时钟 | 行时序（像素数） | | | | | 帧时序（行数） | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | （MHz） | a | b | c | d | e | o | p | q | r | s |
| 640x480@60 | 25.175 | 96 | 48 | 640 | 16 | 800 | 2 | 33 | 480 | 10 | 525 |
| 640x480@75 | 31.5 | 64 | 120 | 640 | 16 | 840 | 3 | 16 | 480 | 1 | 500 |
| 800x600@60 | 40.0 | 128 | 88 | 800 | 40 | 1056 | 4 | 23 | 600 | 1 | 628 |
| 800x600@75 | 49.5 | 80 | 160 | 800 | 16 | 1056 | 3 | 21 | 600 | 1 | 625 |
| 1024x768@60 | 65 | 136 | 160 | 1024 | 24 | 1344 | 6 | 29 | 768 | 3 | 806 |
| 1024x768@75 | 78.8 | 176 | 176 | 1024 | 16 | 1312 | 3 | 28 | 768 | 1 | 800 |
| 1280x1024@60 | 108.0 | 112 | 248 | 1280 | 48 | 1688 | 3 | 38 | 1024 | 1 | 1066 |
| 1280x800@60 | 83.46 | 136 | 200 | 1280 | 64 | 1680 | 3 | 24 | 800 | 1 | 828 |
| 1440x900@60 | 106.47 | 152 | 232 | 1440 | 80 | 1904 | 3 | 28 | 900 | 1 | 932 |

## 3.4 摄像头 OV7670 显示

实验最初设想是从摄像头 OV7670 处采集实时画面并将处理后的图像显示在显示屏上，因此对摄像头显示模块也做了深入的研究，限于时间有限，未能将摄像头调试到最好的状态，因此在最后的项目中采用了 ROM 配置一张图的方式来展示处理成果。此处简要的介绍一下驱动 OV7670 的方法。

### 3.4.1 I2C 串口与寄存器配置

OV7670 内有两百多个寄存器参数，有的是只读的产品信息，绝大多数则是摄像头性能的参数设置，如数据输出格式、白平衡、色彩增益等等，一个好的寄存器配置可以很好的还原拍摄结果，相反寄存器一旦配置出现重大错误，则很可能不能得到有效的结果。

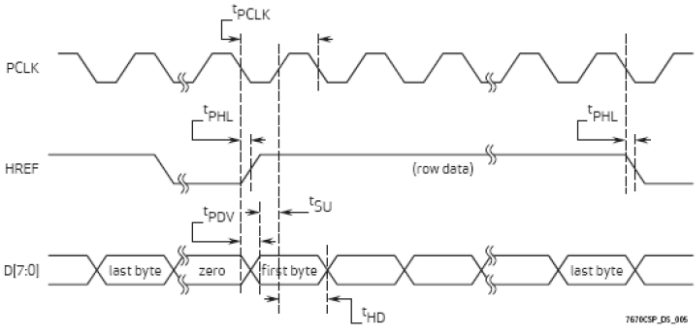
寄存器配置是在驱动摄像头最开始实现的，可看作初始化过程，这一过程的数据传输依靠的是 I2C 数据串口，具体的 I2C 数据传输协议以及其 Verilog 语言实现在此不做赘述，可以自行寻找源码，值得注意的是 I2C 数据串口的结构是 OD 门，因此在 OV7670+Nexys 4 的环境下，需要自行外接上拉电阻。

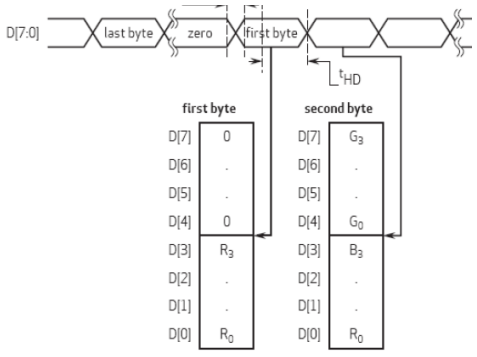
### 3.4.2 HS 和 VS 时序控制与串行数据

摄像头的扫描时序原理和 VGA 是一样的，不同的只是 VGA 需要自行产生时序图并输出给显示屏，而摄像头作为外设，其 HS 和 VS 信号是摄像头的输出，在 FPGA 端只需要将其作为输入进行计数和相应的存储控制即可。

由于 OV7670 的数据输出接口只有 8 位，因此在使用 RGB444 或 RGB565 格式输出视频信息的时候，使用了时分复用的方法，分两个时钟周期共 16bit 来传输 1 个像素信息，即摄像头的像素时钟输出是外部主时钟输入的一半。

OV7670 的输出时序如下两图所示。



3.4.3 数据储存方式

由于是一头输入一头输出的方式对数据作动态实时处理，很自然地想到了双口 RAM 和 FIFO。具体配置方法与上文的 IP 核配置类似，不做多讲。

## 3.5 二值化

设置一个 threshold，作为二值化的阈值，先按一定比例将 RGB 图转成灰度图，对于灰度图每一个像素点的灰度值与 threshold 进行比较，大于等于就将 RGB 值重新赋值为 255，小于则赋值为 0，然后再将新的二值化灰度图转为 RGB 图，将串行输入每个像素点的 12 位数据传出。

判定代码如下。

```
output_data <= (input_data > thresh)? 12'd4095 : 12'd0;
```

**腐蚀与扩展**

腐蚀：黑色部分增多，白色部分减少。

扩展：白色部分增多，黑色部分减少。

上述功能可以通过设置两个按钮开关实现，一个用于控制腐蚀程度的加深，另一个用于控制扩展程度的加深。实现的原理为改变 threshold，通过按钮触发来改变 threshold 的取值，每触发一次就+1 或者-1。

对参数的控制也很简单：

```
else if ((up)&&(thresh<12'b1111_1111_1111))   thresh=thresh+12'b000100010001;
else if ((down)&&(thresh>12'b0000_0000_0000))    thresh=thresh-12'b000100010001;
```

## 3.6 老照片

老照片滤镜的实现方法可以概括为，加大 red、green 色层的权重，减小 blue 色层的权重，可以使滤镜下的照片显得泛黄与古旧。

具体参数的选取见下图：

```
rR=(input_data[11:8]*296+input_data[7:4]*579+input_data[3:0]*149)/1024;
rG=(input_data[11:8]*297+input_data[7:4]*584+input_data[3:0]*143)/1024;
rB=(input_data[11:8]*272+input_data[7:4]*534+input_data[3:0]*131)/1024;
output_data[11:8]=(3*rR+input_data[11:8])/4;
output_data[7:4]=(3*rG+input_data[7:4])/4;
output_data[3:0]=(3*rB+input_data[3:0])/4;
```

其中，参数的选取可以做适当的修改以取得更好的效果，甚至可以取得其他的滤镜效果。但是值得注意的是，输出数据一定是 444RGB，运算中要防止数据溢出的可能，否则会出现异常色块的现象。

## 3.7 负片

负片滤镜类似于早年使用老式相机的胶卷底片。实现方法为将 red、green、blue 的三个色层的取值由 x 变为 15-x。选择 15 是因为 4 位色彩最大值即为$(1111)_2 = 15$。

### 3.8 色相变换

串行输入的数据有 12 位，原本是每 4 位顺序对应 RGB 三个色层，输出到 VGA 上正确显示。现在，为了改变图像的色相，将 12 位数据对应的色层进行交换，由简单的排列组合知识可得一共有 5 种交换方式：RBG、GRB、GBR、BRG、BGR，输出到 VGA 上就能看到不同色相的图像，给人眼前一亮。举例说明代码的实现：

```
output data <= {input data[11:8], input data[3:0], input data[7:4]};
```

### 3.9 数据选择模块

数据选择模块接收从上述几个数据处理模块的输出，实现根据开关控制来输出不同的数据处理结果到 VGA 显示模块。由于所有的处理均是基于单像素的串行数据，同时我们没有考虑两种及以上的处理同时进行的状态，因此并联多个处理模块是完全可行的。从而数据选择模块可以视作在并联的多个数据处理模块后串联的一个数据选择器。这里，数据选择器的编写中有一个处理的优先级，即某些功能对应的开关同时为有效电平时，只显示优先级高的处理结果，以此解决多个开关信号冲突的问题。

## 四 硬件测试结果与视频展示

硬件测试效果符合预期，图像的色彩和视觉效果变化非常显著，实现了相应的功能。具体的测试结果可见附件视频。

## 五 实验总结

本次实验作为电子线路设计与测试实验的最后课程设计，极大的考验和锻炼了我们的硬件编程能力。我们组花了很多时间在摄像头的调试和显示上，但是最终的效果一直不够完美，浪费了不少研究图像处理算法的时间。但是我们最终实现了最初的基本设想，完成了设计任务。

总的来讲，本实验综合了 VGA 时序控制与动态扫描显示，FPGA 的 IP 核配置与 ROM、RAM 等储存器的使用，I2C 传输协议，图像的简单处理、像素 RGB 色彩计算与转换等等多方面的知识。最终实现了一个基于 FPGA 平台的简易图像处理系统，过程中经历了各种软硬件的调试与测试，取得了非常成功的结果。当然，这套系统暂时还非常的简易，值得提升的地方还有很多，可以增加的功能也同样不少，在今后的学习中我们也可以进一步的完善与改进这一系统。

## 六 附代码

（代码见电子档）