



**汇编与接口课程设计**

团队实验报告

|  |  |
| --- | --- |
| 学 院 | 计算机学院 |
| 专 业 | 计算机科学与技术 |
| 指导老师 | 王娟 |
| 组 长 | 胡梦君 |
| 组 员 | 郭峰、崔墨迪、周一鸣 |
| 组长联系方式 | 15908267869 |

二O二二 年 八 月

目 录

[第一章 项目简述 1](#_项目简述)

[第二章 组员分工 1](#_组员分工)

[第三章 设计目的 1](#_设计目的)

[第四章 设计环境 1](#_设计环境)

[第五章 设计原理及内容 2](#_设计原理及内容)

[5.1 数据通路 2](#_数据通路)

[5.2 控制逻辑 2](#_控制逻辑)

[第六章 设计与实现 3](#_设计与实现)

[第七章 测试 5](#_测试)

[第八章 问题及解决方法 6](#_问题及解决方法)

[第九章 心得体会及总结 6](#_心得体会及总结)

[第十章 参考文献有价值的资源推荐 6](#_参考文献有价值的资源推荐)

# 项目简述

集成CPU主机模块和VGA外设控制接口模块，设计汇编程序，通过CPU控制VGA外设。

# 组员分工

小组分工见表2.1 小组分工。

|  |  |
| --- | --- |
| **组员** | **分工任务** |
| 组长：胡梦君 | * 接口设计 |
| 组员：郭峰 | * 图形绘制 |
| 组员：崔墨迪 | * CPU与接口对接 |
| 组员：周一鸣 | * 上板测试 |

表2.1 小组分工

# 设计目的

根据精工板资源，选择设计并完成一种接口控制器，如VGA控制器、LCD、UART、蓝牙等。

# 设计环境

|  |  |
| --- | --- |
| 操作系统 | Windows10 |
| 编程语言 | Verilog HDL |
| EDA工具 | Vivado2019.2 |
| 汇编语言 | MIPS |
| 汇编程序编辑器 | mars4\_5 |

# 设计原理及内容

## 数据通路

由于沿用了流水线CPU设计项目中完成的CPU模块，以下仅展示vga外设和confreg外设控制两模块的数据通路。

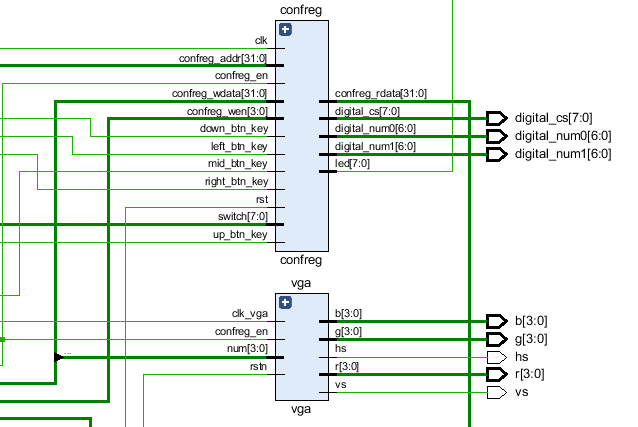


图5.1.1 VGA外设和控制模块数据通路

## 控制逻辑

confreg模块和vga模块采用时序逻辑实现。

vga模块的控制信号如下：

* **confreg\_en**：外设写使能。
* **num**：需要显示的数字。
* **hs**：vga接口的水平同步信号。
* **vs**：vga接口的垂直同步信号。
* **r, g, b**：颜色控制信号。

confreg模块的控制信号如下：

* **confreg\_en**：外设写使能。
* **confreg\_wdata**：CPU向外设传输的数据。
* **confreg\_addr**：CPU写外设的地址。
* **digital\_num0**：数码管0到3位显示数字。
* **digital\_num1**：数码管4到7位显示数字。
* **digital\_cs**：数码管八位片选信号。

# 设计与实现

将VGA接口控制器与流水线CPU整合，CPU在执行访存模块dmemreq后会向同时上层输出外设写使能与待写入的数据。经转换器bridge转换后进入外设控制器confreg并在其中完成数据过滤，最后成为VGA的输入。

VGA模块接口如下所示。

module vga (

input clk\_vga,

input rstn,

input [3:0] num,

input confreg\_en,

output hs,

output vs,

output [3:0] r,

output [3:0] g,

output [3:0] b

);

VGA模块模拟十三段数码管，根据CPU输入的数字，采用时序逻辑控制，向显示器发送同步信号，对十三个区域的像素RGB值进行赋值。如果发送的像素所在位置的数码管是亮的，RGB = (r,g,b)，否则RGB = (0,0,0)。部分代码如下所示。

always @ (posedge clk\_vga) begin

if (h\_cur > 700 && h\_cur < 900 && v\_cur > 50 && v\_cur < 150 ) begin

if (light[0]) begin

reg\_r <= `RED\_R;

reg\_g <= `RED\_G;

reg\_b <= `RED\_B;

end

else begin

reg\_r <= `BLACK\_R;

reg\_g <= `BLACK\_G;

reg\_b <= `BLACK\_B;

end

end

else if (h\_cur > 600 && h\_cur < 700 && v\_cur > 150 && v\_cur < 350 )

... ...

... ...

... ...

else if (h\_cur > 900 && h\_cur < 1000 && v\_cur > 650 && v\_cur < 750 ) begin

if (light[5] || light[6]) begin

reg\_r <= `RED\_R;

reg\_g <= `RED\_G;

reg\_b <= `RED\_B;

end

else begin

reg\_r <= `BLACK\_R;

reg\_g <= `BLACK\_G;

reg\_b <= `BLACK\_B;

end end

else begin

reg\_r <= `BLACK\_R;

reg\_g <= `BLACK\_G;

reg\_b <= `BLACK\_B;

end

end

数码管译码将输入的4位二进制数转换为对应数码管的控制信号，1表示亮，0表示暗。代码如下所示。

assign light = (confreg\_en == 0) ? light :

(num == 0) ? 7'b1110111 :

(num == 1) ? 7'b0100100 :

(num == 2) ? 7'b1011101 :

(num == 3) ? 7'b1101101 :

(num == 4) ? 7'b0101110 :

(num == 5) ? 7'b1101011 :

(num == 6) ? 7'b1111011 :

(num == 7) ? 7'b0100101 :

(num == 8) ? 7'b1111111 :

(num == 9) ? 7'b1101111 :

(num == 10) ? 7'b0111111 :

(num == 11) ? 7'b1111010 :

(num == 12) ? 7'b1010011 :

(num == 13) ? 7'b1111100 :

(num == 14) ? 7'b1011011 :

(num == 15) ? 7'b0011011 :

7'b0000000;

# 测试

用于测试的汇编代码如下。

    .org 0x0

    .set noat

    .set noreorder

    .set nomacro

    .global \_start

\_start:

    lui $1, 0x1faf

    ori $1, $1, 0x00008000

    sw  $t0, 0x0($1)

loop:

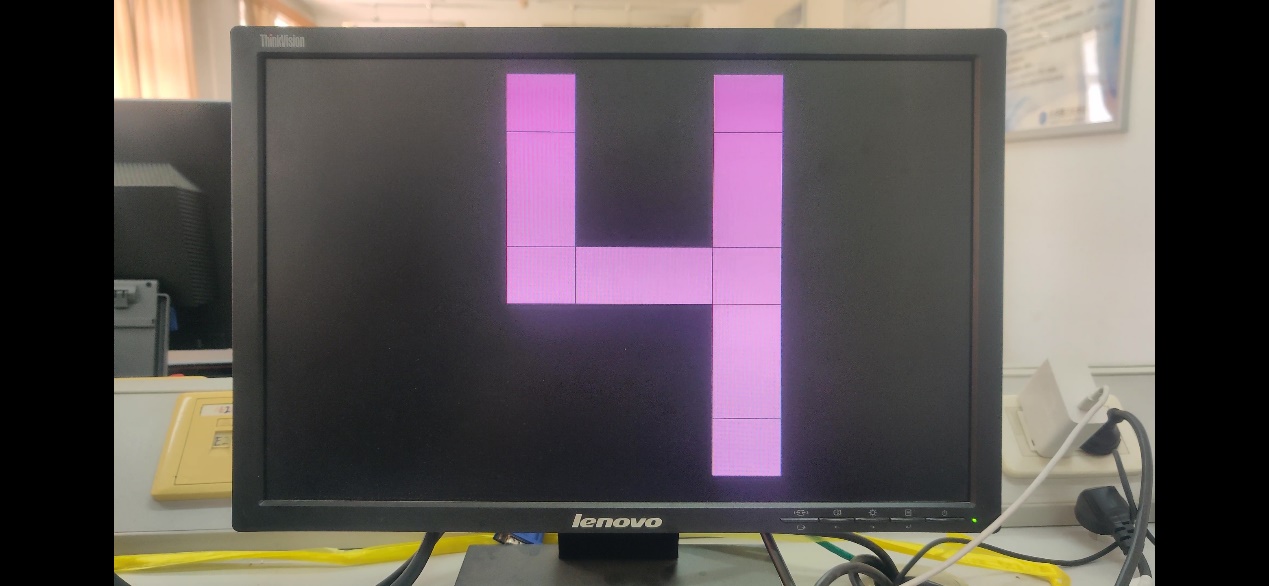
    addiu $t0, $t0, 0x00000001

    sw $t0, 0x0($1)

    j loop

测试代码先将数码管外设测试地址保存至1号寄存器，并将初始值写入该地址所指向的内存单元，然后将该值循环递增，以此在显示器上循环显示十六进制数0~F。

显示效果如下图所示。



# 问题及解决方法

最初VGA接口仿真结果正确，但上板测试时显示器无法显示正确结果。经检查发现是时钟频率问题导致，应该根据所使用的分辨率计算对应的VGA时钟信号。创建一个IP核来实现时钟信号的转换，即可解决这个问题。

# 心得体会及总结

通过设计和实现简单的VGA接口，我们为之前设计的流水线CPU提供了简单的输出，整体初步实现了简单的计算系统。在本次实验中，我们对计算机如何进行接口控制有了更深入的理解，加深了对接口技术的理解和认识。

# 参考文献有价值的资源推荐

无