实验平台说明

特征	说明
操作系统版本和系统类型	Windows 10 64 bit
软件版本	Quartus ii13.0 及 IAR Embedded Workbench
验证方式	PC 验证及硬件验证

目录

一． 系统概述

1.1 概述

本课程设计是基于 ARM 嵌入式系统的 PI 计算与显示，其中主要包括硬件系统与软件系统，软件主要包括：PI 计算算法，驱动底层及按键检测，定时。

实现的功能为，在显示系统上显示 100 位 PI。

1.2 方案选择

在上本课程之前，我有幸接触过几类单片机，如 ST 公司 STM32F1 系列，F4 系列，NXP 公司：MK60DN512VLQ10，MK66FX1MOVLQ18，TI 公司：MPS430 及 DSP 芯片：TMS320F28335。非常可惜这些芯片都在学校，而且目前由于马上要进行考研复习，没有购买的需求。

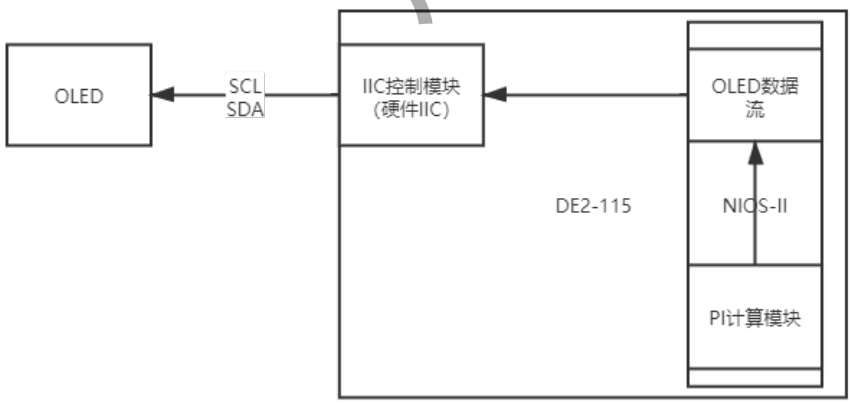
本课程设计原本应在 NIOS II 上进行测试并实验，本课程设计中我使用 DE2-115 为底层硬件，NIOSII 为软件架构来进行设计，并且由 K66 底层写出测试算法交予朋友来进行 K66+OLED 平台上的测试。在此也感谢自动化 1805 班的施阳学弟帮我进行测试，并且指正了一些错误。

二． 硬件系统

2.1 系统架构

硬件原料：DE2-115+OLED

系统结构图：



2.2 IIC 控制系统

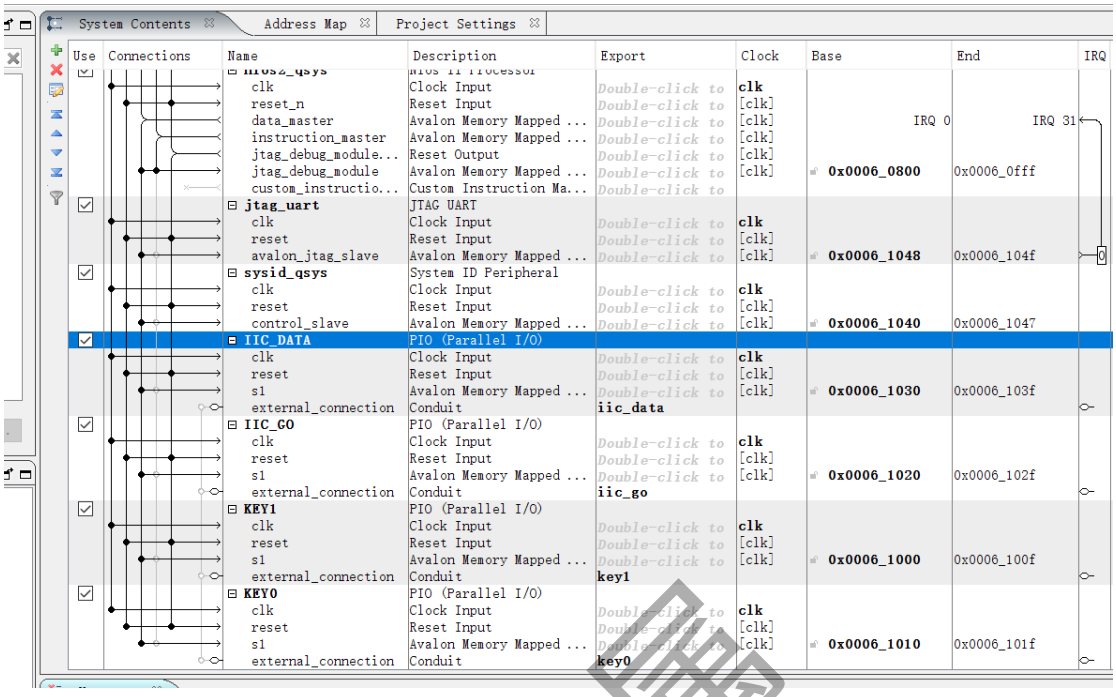
这里需要采用基于 verilog 语言的硬件 IIC 来进行模块设计，我查寻到 DE2-115 的光盘例程 TV_control 中有 IIC 的单独文件，所以这里采用此 iic_controller.v 和 I2C_Confog.v 现成模块使用。

```
// ============================================================================
//
// Major Functions:i2c controller
//
// ============================================================================
//
// Revision History :
// ============================================================================
//   Ver  :| Author            :| Mod. Date :| Changes Made:
//   V1.0 :| Joe Yang          :| 05/07/10  :|      Initial Revision
// ============================================================================
module I2C_Controller (
    CLOCK,
    I2C_SCLK,//I2C CLOCK
    I2C_SDAT,//I2C DATA
    I2C_DATA,//DATA:[SLAVE_ADDR,SUB_ADDR,DATA]
    GO,      //GO transfor
    END,     //END transfor
    W_R,     //W_R
    ACK,      //ACK
    RESET,
    //TEST
    SD_COUNTER,
    SDO
);
```

```
module I2C_Config (  // Host Side
                     input sys_clk,
                     input sys_rst_n,
                     input[23:0] I2C_DATA,
                     input GO,
                     // I2C Side
                     output I2C_SCLK,
                     inout I2C_SDAT
                     );
```

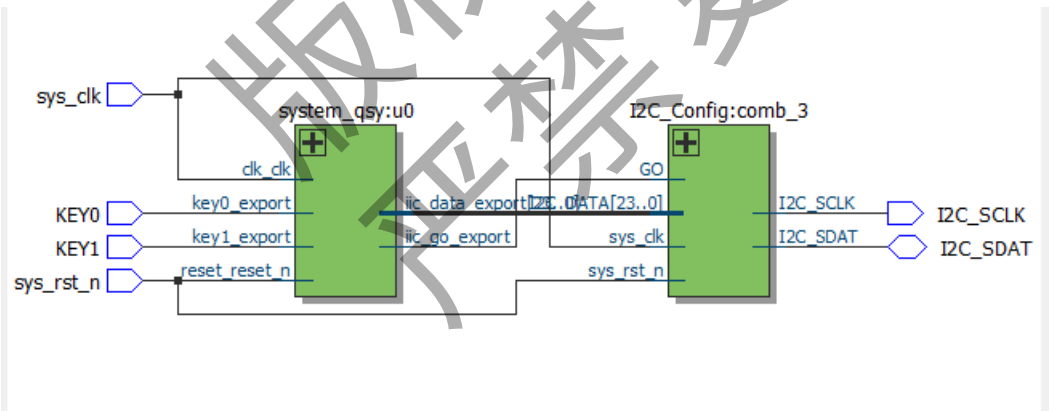

2.3NIOS II 系统

使用 Qsys 来创建 NIOS II 系统，其中 NIOS II 系统包括：24 位端口 IIC_DATA 一位端口 IIC_GO,一位端口 KEY1，一位端口 KEY0，片上 ROM，片上 RAM，SYSID , JTAG_UART , NIOS II TIME0 等，如图：



2.4 顶层文件

创建 qrs_curriculum_design.v 的顶层文件,主要为了连接 IIC_CONTROLL 和 NIOS II 的系统:
如图为 RTL 级视图:



如图为.v 文件,

```
module qrs_curriculum_design(
input sys_clk,
input sys_rst_n,
inout I2C_SDAT,
input KEY0,
input KEY1,
output I2C_SCLK
);

reg   [23:0]   mI2C_DATA;
reg         mI2C_GO;

wire     mI2C_END;
wire     mI2C_ACK;
```

```
    system_qsy u0 (
        .clk_clk         (sys_clk),         //      clk.clk
        .reset_reset_n   (sys_rst_n),   //    reset.reset_n
        .iic_data_export (mI2C_DATA), // iic_data.export
        .iic_go_export   (mI2C_GO),   //   iic_go.export
        .key1_export     (KEY1),     //     key1.export
        .key0_export     (KEY0)      //     key0.export
    );

    I2C_Config (   // Host Side
                   .sys_clk(sys_clk),
                   .sys_rst_n(sys_rst_n),
                   // I2C Side
                   .I2C_SCLK(I2C_SCLK),
                   .I2C_SDAT(I2C_SDAT),
                   .I2C_DATA(mI2C_DATA),
                   .GO(mI2C_GO)
    );
```

如有 DE2-115 实物,则只需匹配引脚并下载即可完成硬件搭配。

三． 软件系统

3.1 PI 计算算法

PI 的计算有很久的历史，通过对高数的学习，一般大学生，可以知道有几种 PI 计算的方法。
分数逼近法。

- 3.1415926535 8979323846 2643383279 5028841971 6939937510 … (from Wikipedia)
- $\pi = 4\left(\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}-\frac{1}{15}+\frac{1}{17}-\frac{1}{19}+\frac{1}{21}-\frac{1}{23}+\frac{1}{25}-\frac{1}{27}+\frac{1}{29}-\frac{1}{31}+\cdots\right)$

使用这种方法来计算 PI，因为我们知道 C 语言的 double 和 float 型数据精度实在有限，而且逼近法对于清楚的 PI 公式计算非常模糊。所以这里不能采用这种方法。
或者马青公式：

$$Machin: \pi = 16\arctan\frac{1}{5} - 4\arctan\frac{1}{239}$$

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9}\cdots$$

$$\pi = 16\times\left(\frac{1}{1\times 5} - \frac{1}{3\times 5^3} + \frac{1}{5\times 5^5} - \frac{1}{7\times 5^7} + \frac{1}{9\times 5^9}\cdots\right) - 4\times\left(\frac{1}{1\times 239} - \frac{1}{3\times 239^3} + \frac{1}{5\times 239^5} - \frac{1}{7\times 239^7} + \frac{1}{9\times 239^9}\cdots\right)$$

马青公式，网上有例程计算并储存 100 位，这里我不采用此种方法是感觉这种方法的计算量比较大。
下面介绍一种非常复杂的模糊算法：

```
#include <stdio.h>
int a=10000, b=0, c=400, d=0, e=0, f[401], g=0,num[100],x=0;

int main()
{
for(;b-c;)
f[b++]=a/5;
for(;d=0,g=c*2;c-=14,num[x]=e+d/a,printf("%.4d",num[x]),x++,e=d%a)
for(b=c; d+=f[b]*a, f[b]=d%--g, d/=g--, --b; d*=b);
return 0;
}
```

这段代码的出处已经鲜为人知，有人说这段代码在 18 届国际模糊 C 语言大赛中获得冠军，也有人说这段代码是某位高中大神所写。

至于原理，网上有这样一句话，当你试图用 for 循环来分析该段代码时，你就会掉入一种模糊的境界，无法分析出原理。在这里不再分析，因为实在是过于复杂。

使用普通的编译器进行编译，并查看结果，可以发现，百度诚不欺我。

3.2 OLED 底层驱动

由于本课程设计使用硬件 IIC，所以只需在软件上进行组合即可，我这里移植中景园电子的 OLED.C 程序。主要程序包含以下：

```
void delay_us(u16 time_us);//延迟1us
void delay_ms(u16 time_ms);//延迟1ms
void I2C_WriteByte(u8 addr, u8 data);//IIC写入
void OLED_WR_Byte(u8 dat,u8 cmd,u8 type);//写寄存器
void OLED_Set_Pos(unsigned char x, unsigned char y);//一个点
void OLED_Display_On(void);//打开显示
void OLED_Display_Off(void);//关闭显示
void OLED_Clear(void);//清屏
void OLED_ShowChar(u8 x, u8 y, u8 chr, u8 type);//在固定位置显示一个字节
u32 oled_pow(u8 m, u8 n);
void OLED_ShowNum(u8 x, u8 y, u32 num, u8 len, u8 size, u8 type);//显示一个数字
void OLED_ShowString(u8 x, u8 y, u8 *chr, u8 type);//显示一个字符串
void OLED_ShowCHinese(u8 x, u8 y, u8 no, u8 type);//显示一个汉字
void OLED_DrawBMP(unsigned char x0, unsigned char y0, unsigned char x1, unsigned char y1, unsigned char BMP[], u8 type);//显示一幅图
void OLED_Init(void);//初始化
void show_pi_ten(int c[100],int i);//显示PI
```

3.3 按键检测及主函数

下面逐句分析主函数：

```
int main()
{
    int a=10000, b=0, c=400, d=0, e=0, f[401], g=0,num[100]={0
},x=0;
    u8 i=0;
    u32 key0,key1;//两个按键值
    OLED_Init();//oled 初始化
    OLED_ShowString(45,3, (u8 *)"Hello from Nios II!",1);//显示
初始化标语
    delay_ms(100);//延迟一会儿等待系统稳定
    while(1)
    {
        if(num[99]==0)//若此时没有计算 PI 则开始计算，否则执行显示
的程序
        {
        for(;b-c;)
        f[b++]=a/5;
        for(;d=0,g=c*2;c-
=14,num[x]=e+d/a,printf("%.4d",num[x]),delay_ms(1),x++,e=d%a)
        for(b=c; d+=f[b]*a, f[b]=d%--g, d/=g--, --b; d*=b);
        }//计算 PI
        else
        {
```

```
            show_pi_ten(num,i);//显示第 10*i 到 10*i+9 的值，即一
行显示十个。
            key0 = IORD_ALTERA_AVALON_PIO_DATA(KEY0_BASE);//采
集 KEY0 状态
            key1 = IORD_ALTERA_AVALON_PIO_DATA(KEY1_BASE);//采
集 KEY1 状态
            if(key0 == 0)//如果 KEY0 按下
            {
                delay_ms(10);//防抖
                if(key0 == 0)//再次检测到 KEY0 按下
                {
                    i++;//系数 I++
                    if(i>=10)i = 10;如果大于 10 则不加
                }
            }
            if(key1 == 0)//检测到 key1 按下
            {
                delay_ms(10);//防抖
                if(key1 == 0)//再次检测到 key1 按下
                {
                    i--;//i--
                    if(i<=0)i = 0;//若小于 0，则等于 0
                }
            }
        }
        delay_ms(5);
    }
  return 0;
}
```

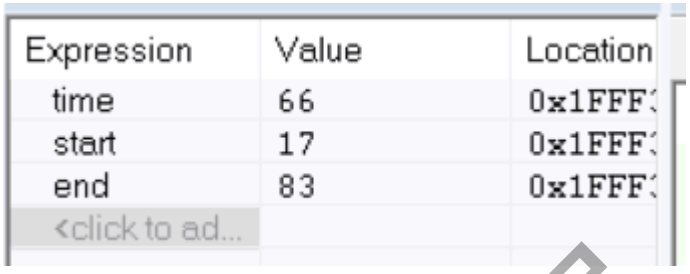
如此便可显示 PI 的值在 oled 上。

四． 验证

在 K66 上进行验证。并且计算程序运行时间：

以下是测试时间程序

```
while(1)
{
    pit_time_start();
    start = pit_time_get();
    for(;b-c;)
        f[b++]=a/5;
    for(;d=0,g=c*2;c-=14,num[x]=e+d/a,x++,e=d%a)
        for(b=c; d+=f[b]*a, f[b]=d%--g, d/=g--, --b; d*=b);
    end = pit_time_get();
    time = end-start;
    pit_close();
    vTaskDelay(1000);
}
```



Expression	Value	Location
time	66	0x1FFF
start	17	0x1FFF
end	83	0x1FFF
<click to ad...		

单位为 us，则运行 66us，可见算法的时间复杂度还是很低的。

五. 实物

视频随在文件夹中，与 NIOS II 程序略有不同，且有一 BUG 还未解决。

六. 心得体会

本次课程设计熟悉了，NIOS II 与 DE2-115 的嵌入式整体开发。令我印象比较深刻还是 FPGA 的硬件特性与 MCU 元件特性的结合。在此之前我所使用的 IIC 的算法多为软件 IIC，延时很长，波形很差，速度较慢，而 FPGA 和 MCU 的结合非常简单，可以使难度均摊在硬件和软件上，我想这就是 FPGA 的优越所在。

最后感谢自动化 1805 班的施阳学弟帮助我完成本课程设计。