

专业: 计算机应用技术

姓名: 杜沈达

学号: SA18168163

日期: 2019年1月2日晚

地点: 一教实验室

课程名称: 可编程逻辑器件原理与应用 实验名称: 课程设计 成绩:

1 简介

我的课程设计所用的 DE2-115 开发板, Quartus Prime17.0 软件, DE2-115 开发板外部有很多外围设备,能够保证足够的使用资源。使用 Platform Designer 制作需要功能的芯片, 然后使用 Quartus 制作 bdf 文件, 之后使用 Nios II for Eclipse17.1 软件编写程序, 再将软硬件都烧录进目标板, 也就实现了所需要的设计,设计的步骤如图 1 所示。

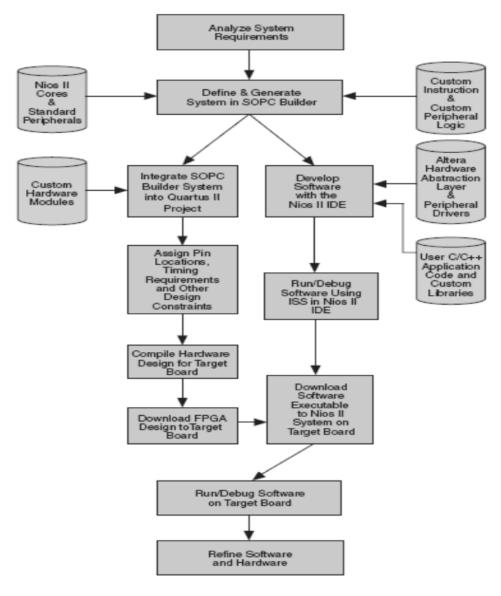


图 1: 设计流程图

2 实现的功能

- 1. 实现 8 个绿色 LED 跑马灯;
- 2. 实现数码管四位每秒自增一计时器;
- 3. 实现数码管一位按下自增一计数器。

3 设计过程

3.1 NIOS 设计

如图 2 所示,先加入 nios ii cpu,进行计算与控制,sdram 作为随机存储器,uart 用 jtag 口,作为调试口与 PC 通信,system id 作为 cpu 标号,作为软硬件的交互,与时间戳一起判断是否匹配,使用一个 1ms,32 位的计时器,使用 PIO 口作为输出引脚,电平触发,对于 LED 采用 8 位 PIO 输出,数码管每个是 7 位输出,按钮 1 位输入。

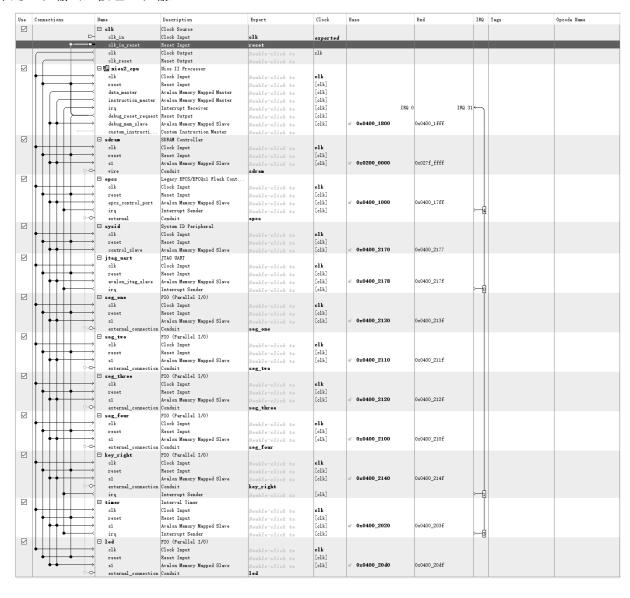


图 2: NIOS II 设计

使用 Avalon Memory Mapped 内存映射总线,连接时钟,置位,读写地址与数据等信号,所有的时钟均采用 $50 \mathrm{MHz}$,在按钮,epcs,jtag 和定时器处设置断点,其中基地址由软件自动分配,完成后 Generate 成.bsf 和 Verilog 语言的文件。

PIO (Parallel I/O) altera avalon pio Details	PIO (Parallel I/O) altera avalon pio Details
Y = 1 =	Y B
Width (1-32 bits):	Width (1-32 bits): 7
Direction: OBidir	Direction: O Bidir
O Input	O Input
◯ InOut	○ InOut
• Output	① Output
Output Port Reset Value: 0x00000000000000000000000000000000000	Output Port Reset Value: 0x00000000000000000000000000000000000
Output Register	Output Register
Enable individual bit setting/clearing	☐ Enable individual bit setting/clearing
Edge capture register	* Edge capture register
Synchronously capture	Synchronously capture
Edge Type: RISING V	Edge Type: RISING
Enable bit-clearing for edge capture register	Enable bit-clearing for edge capture register
▼ Interrupt	* Interrupt
Generate IRQ	Generate IRQ
IRQ Type: LEVEL V	IRQ Type: LEVEL >
	LEVEL V
Level: Interrupt CPU when any ummasked I/O pin is logic true Edge: Interrupt CPU when any ummasked bit in the edge-capture register is logic true. Available when synchronous capture is enabled	Level: Interrupt CPU when any ummasked I/O pin is logic true Bdge: Interrupt CPU when any ummasked bit in the edge-capture register is logic true. Available when synchronous capture is enabled
Test bench wiring	Test bench wiring
Hardwire PIO inputs in test bench	Hardwire PIO inputs in test bench
Drive inputs to field.: 0x00000000000000000000000000000000000	
52100 Inputs to ITeru [3x000000000000000000000000000000000000	Drive inputs to field.: 0x00000000000000000000000000000000000
System: nisoii Path: timer	PIO (Parallel I/O)
Interval Timer	PIO (Parallel I/O) altera svalon pio Details
Interval Timer	altera avalon pio Details
Interval Timer	D-+11-
Interval Timer altera avalon timer Details	Basic Settings Width (1-32 bits):
Interval Timer altera avalon timer Timeout period	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir
Interval Timer altera avalon timer Timeout period Period: Units: ms	Basic Settings Width (1-32 bits):
Interval Timer alters avalon timer Timeout period Period: 1 Units: ms Timer counter size	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir
Interval Timer altera avalon timer Timeout period Period: Units: ms	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir Input
Interval Timer altera avalon timer Timeout period Period: 1 Units: us Timer counter size Counter Size: 32	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir Input Inout Output
Interval Timer altera avalon timer Timeout period Period: Units: ms Timer counter size Counter Size: 32 >	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir Input Inout
Interval Timer altera avalon timer Timeout period Period: Units: Ins Timer counter size Counter Size: 32 Registers In No Start/Stop control bits	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir Input Inout Output
Interval Timer altera avalon timer Timeout period Period: Units: Ins Timer counter size Counter Size: 32 > Registers Is Start/Stop control bits Fixed period	Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Ins Timer counter size Counter Size: 32 Registers In No Start/Stop control bits	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir Input Inout Output Output Port Reset Value: Data Data Data Data Data Data Data Dat
Interval Timer alters avalon timer Timeout period Period: Units: ms Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: 1 Units: Ins Timer counter size Counter Size: 32 Registers Ino Start/Stop control bits Fixed period Readable snapshot Output signals	altera avalon pio Basic Settings Width (1-32 bits): Direction: Bidir Input Inout Output Output Port Reset Value: Dacooooooooooooooooooooooooooooooooooo
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera svalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: 1 Units: Ins Timer counter size Counter Size: 32 Registers Ino Start/Stop control bits Fixed period Readable snapshot Output signals	altera svalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera svalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings Width (1-32 bits): 1 Direction: Bidir Input Output Output Port Reset Value: 0x00000000000000 Output Register Enable individual bit setting/clearing Edge capture register Synchronously capture Edge Type: FALLING
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera svalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings
Interval Timer altera avalon timer Timeout period Period: Units: Is Timer counter size Counter Size: 32 Registers No Start/Stop control bits Fixed period Readable snapshot Output signals System reset on timeout (Watchdog)	altera avalon pio Basic Settings

图 3: 各个连接部件参数 (1)LED 灯;(2) 数码管;(3) 定时器;(4) 按钮

3.2 Quartus 的.bdf 文件绘制和引脚分配

3.2.1 .bdf 文件绘制

如图 4 所示,由 inclk 时钟信号经过 PLL 锁相环输入给 nios ii cpu,将其 reset 位接 vcc,即不置位,其余的管角均右键点击 nios ii cpu 点击 Generate Pins for Symbol Port 产生。

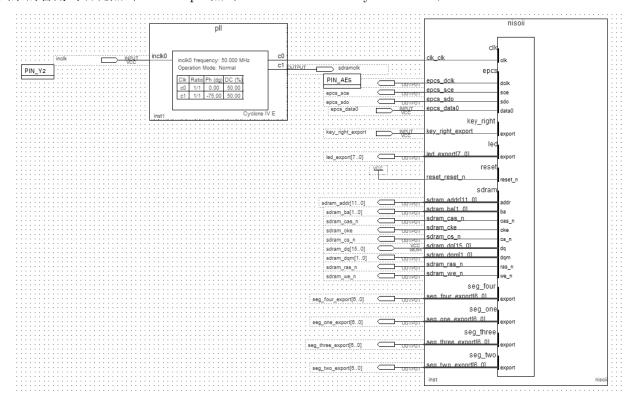


图 4: bdf 文件绘制

3.2.2 引脚分配

参考 DE2-115 的用户手册,对于 8 个绿色 LED 灯的引脚分配如图 5 所示,

LEDG[0]	PIN_E21	LED Green[0]	2.5V	LEDRO LEDRO G19
LEDG[1]	PIN_E22	LED Green[1]	2.5V	F19 LEDG1 LEDG1 LEDG1
LEDG[2]	PIN_E25	LED Green[2]	2.5V	LEDG2 LEDG2 LEDG2
LEDG[3]	PIN_E24	LED Green[3]	2.5V	LEDGS V LEDGS
LEDG[4]	PIN_H21	LED Green[4]	2.5V	LEDR14 LEDR14 F15 Cyclone IV G20
LEDG[5]	PIN_G20	LED Green[5]	2.5V	LEDG15 ALLEDG15 G15 G22 LEDG8
LEDG[6]	PIN_G22	LED Green[6]	2.5V	LEDG7 LEDG7 LEDG8
LEDG[7]	PIN_G21	LED Green[7]	2.5V	LEDRI7 LEDRI7 H15 F17
LEDG[8]	PIN_F17	LED Green[8]	2.5V	÷ ÷

图 5: (1)LED 管脚分配; (2)LED 与 FPGA 连接

对于 4 个 7 段数码管的管脚分配可以由图 6 得到, DE2-115 共有 8 个数码管, 在这里只使用了四个, 作为控制计数器和计时器使用。其中 Key 按钮和其他管脚的分配均可参考用户手册, 或者从已经分配好的 Excel 文件中拷贝得到, 这样更不容易出错, 也大大减小了工作量。

最终的引脚分配如图 7 所示。

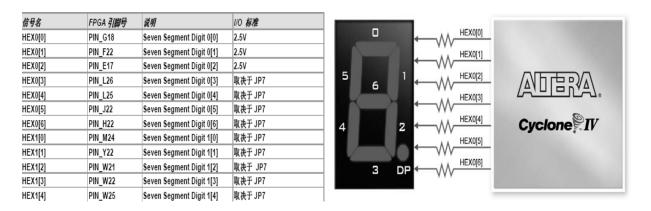


图 6: (1) 数码管管脚分配 (截取部分); (2) 数码管与 FPGA 连接

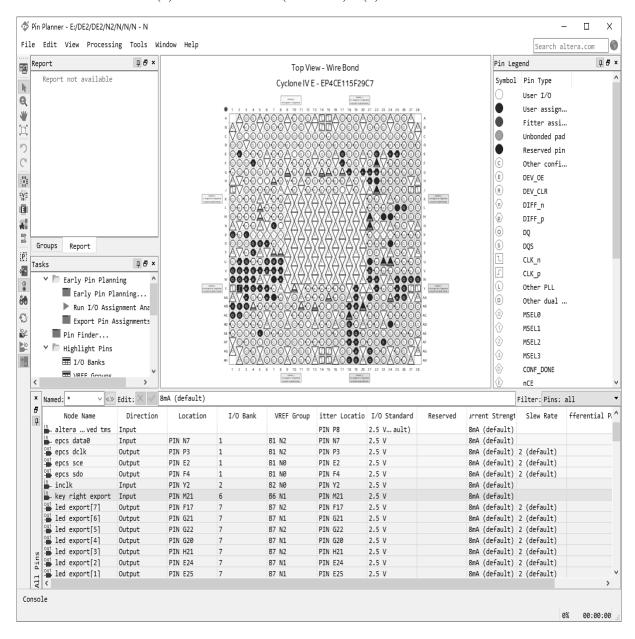


图 7: 最终的引脚分配

3.3 软件编写

在完成了硬件部分的设计之后,要进行的就是软件的设计,从 Platform Designer 的 Tool 里面打开 NIOS II Software Build Tool for Eclipse,选择工作空间,新建一个 NIOS Application and BSP from Template,选择 sopc information 文件。

之后进入软件的编写,首先定义一通用输入输出的结构体 PIO_STR ,包括基地址数据,方向,中断和边沿触发。又定义一定时器寄存器结构体。使用更好理解的 SEG_ONE 等代替数据所在的基地址,设置数码管的 1 9 显示数组。用 Display 函数进行显示输出,在主函数中写一个死循环,数码管每秒加 1,直到碰到按钮进行中断,令对应的控制的数码管加 1,LED 灯的控制完全自己进行。也就是每秒 LED 的寄存器变化一次,下一个灯开始亮,如此循环往复。具体的程序如下:

```
#include <stdio.h>
1
2
  #include "system.h"
3
  #include "sys/alt irq.h"
4
  #include "sys/alt_alarm.h"
  #include "altera_avalon_pio_regs.h"
5
   #include "altera_avalon_timer_regs.h"
6
   #include "unistd.h"
7
8
   #include "alt_types.h"
9
10
   typedef struct
11
12
                    unsigned long int DATA;
                    unsigned long int DIRECTION;
13
14
                    unsigned long int INTERRUPT_MASK;
                    unsigned long int EDGE CAPTURE;
15
16
   }PIO_STR;
17
                            //定时器寄存器
18
   typedef struct
19
20
                    unsigned long int STATUS;
                    unsigned long int CONTROL;
21
22
                    unsigned long int PERIOD L;
                    unsigned long int PERIOD H;
23
24
                    unsigned long int SNAP_L;
25
                    unsigned long int SNAP_H;
26
   }TIMER_STR;
27
28
  #define SEG ONE
                                ((PIO_STR*)SEG_ONE_BASE)
  #define SEG TWO
                                ((PIO STR*)SEG TWO BASE)
29
   #define SEG_THREE
                                ((PIO_STR*)SEG_THREE_BASE)
30
31
  #define SEG_FOUR
                                ((PIO_STR*)SEG_FOUR_BASE)
  #define TIMER
                                ((TIMER_STR*)TIMER_BASE)
                                                            //指向定时器结构体
32
  #define KEY_RIGHT
                                ((PIO_STR*)KEY_RIGHT_BASE)
```

```
34
              unsigned char table [] = \{0 \times 0.0 \times 10^{10}, 0 \times 10^{
35
              unsigned char second = 0;
36
              unsigned char minute = 0;
37
38
              unsigned char hour = 0;
39
              alt_u8 led8 = 0x02;
40
                                                                                                                      // 0000 0010
                                                                                                                  // 0000 0000
41
              alt_u8 dir = 0;
               volatile int i;
42
43
              void ISR_Timer(void *context, alt_u32 id) //定时器中断服务函数
44
45
46
                                                                                    second++;
47
                                                                                    TIMER->STATUS &= 0x00; //清中断标志位
48
              //显示函数
49
              void display (unsigned char hour, unsigned char min, unsigned char sec)
50
51
52
                                                                                    unsigned char ge, shi;
                                                                                     ge = sec \% 10;
53
                                                                                     shi = sec/10;
54
55
                                                                                   SEG ONE \rightarrow DATA = 0 \times ff;
56
                                                                                   SEG_ONE -> DATA = table [ge];
57
                                                                                     usleep (1000);
58
59
60
                                                                                   SEG_TWO \rightarrow DATA = 0 x ff;
                                                                                   SEG TWO -> DATA = table [shi];
61
                                                                                     usleep (1000);
62
63
                                                                                     ge = min\%10;
64
                                                                                     shi = min/10;
65
66
67
                                                                                   SEG THREE \rightarrow DATA = 0 \times ff;
68
                                                                                    SEG_THREE -> DATA = table [ge];
                                                                                     usleep (1000);
69
70
                                                                                   SEG_FOUR \rightarrow DATA = 0xff;
71
                                                                                   SEG_FOUR -> DATA = table [shi];
72
73
                                                                                     usleep (1000);
74
75
76
```

```
int main(void)
                      //主函数
77
78
79
                      //注册中断
                      alt_irq_register(TIMER_IRQ,(void*)TIMER_BASE,ISR_Timer);
80
                      //初始化中断周期,周期为一秒
81
82
                      TIMER \rightarrow PERIOD_L = 50000000\&0 x ffff;
                      TIMER -> PERIOD_H = 500000000>>16;
83
                      TIMER \rightarrow CONTROL = 0 \times 07; // 启动定时器并开中断
84
85
    \mathbf{while}(1)
86
87
         {
    //seg
88
                      if(second > 59)
89
90
                                         second = 0;
91
92
                                         minute++;
93
                       if (minute > 59)
94
95
96
                                         minute = 0;
97
                                         hour++;
98
                       if (hour > 23)
99
100
                               {
101
                                         hour = 0;
102
103
104
                      display (hour, minute, second);
105
106
                       /* i f (KEY_LEFT -> DATA == 0) // 按键控制
107
                                         usleep (20000);
108
109
                                         hour++;
                      }*/
110
111
                       if(KEY_RIGHT \rightarrow DATA == 0)
112
                       {
113
                                         usleep (20000);
114
                                         minute++;
             }
115
116
    // led
117
                      //led=1000 0000/0000 0001时候,这个if语句都会执行
118
                       if (led8 & 0x81)
119
```

```
{
120
                          dir = (dir ^0x01); //1
121
122
123
                if (dir) //1
124
                          led8 = led8 >> 1; //LED右移动显示
125
126
                     }
                    else
127
128
                    {
129
                          led8 = led8 << 1; //LED左移动显示
130
                    }
131
132
                    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, led8); //赋值
133
134
                     i = 0;
                        while (i <250000) //延时
135
136
                             i++;
        }
137
138
                    return 0;
139
```