中国科学技术大学
University of Science and Technology of China

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。

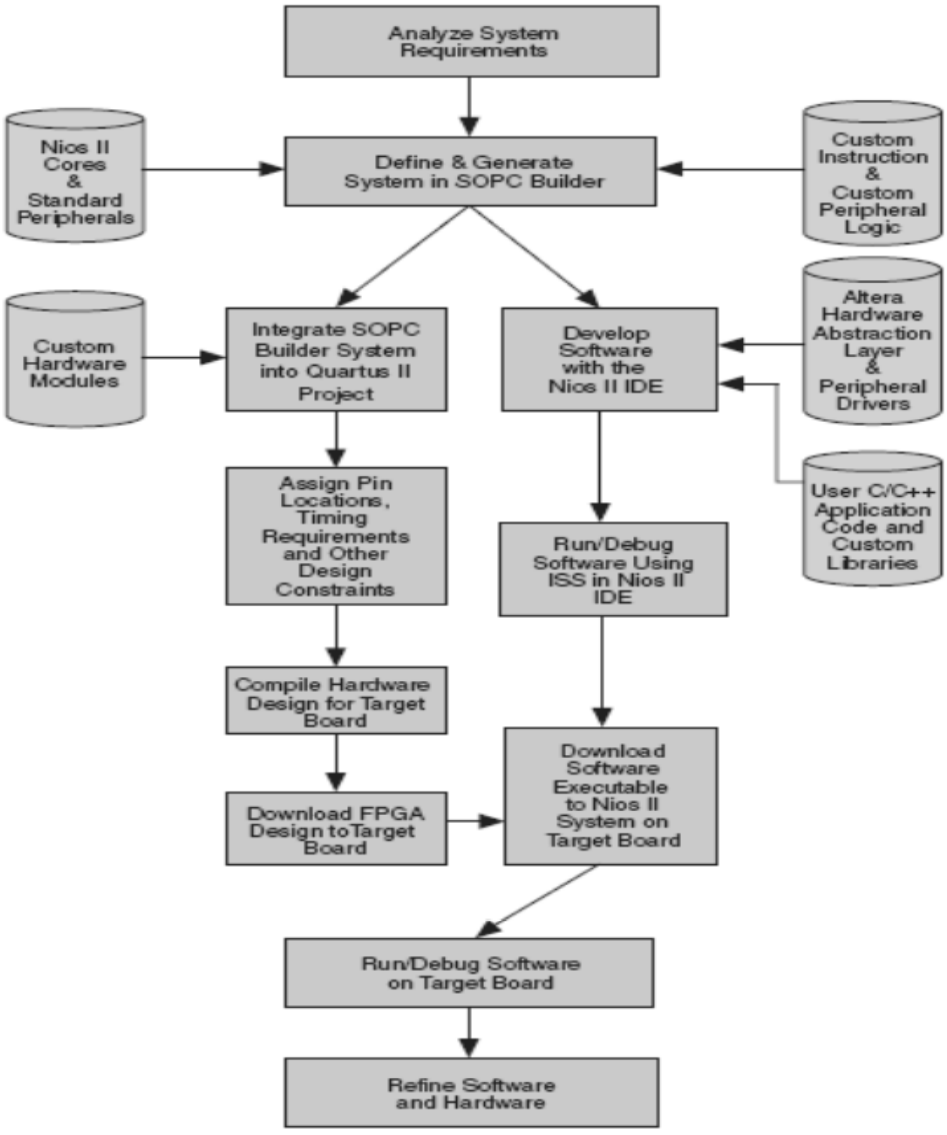

图 1: 设计流程图

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。

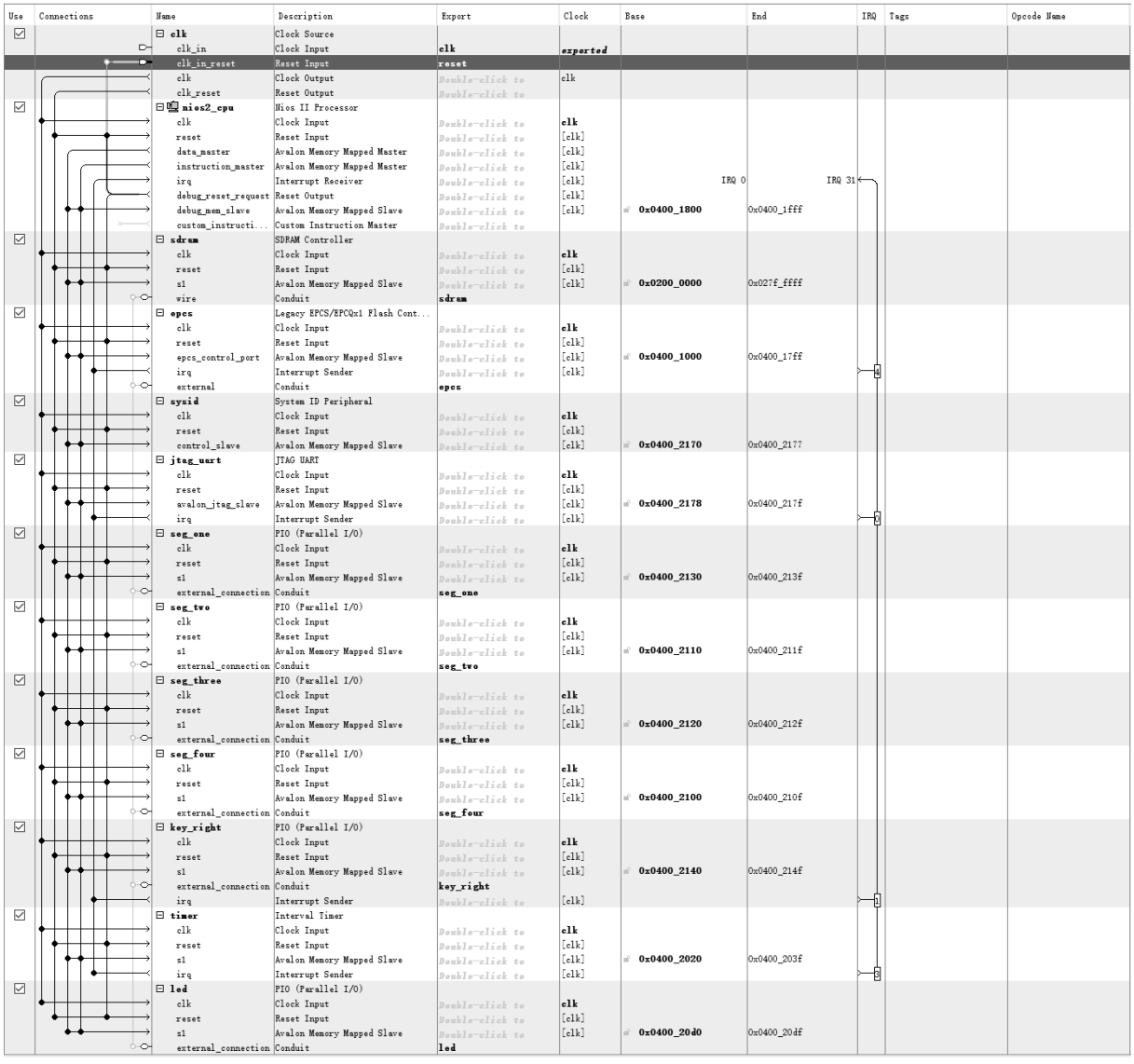

图 2: NIOS II 设计

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 50MHz，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。

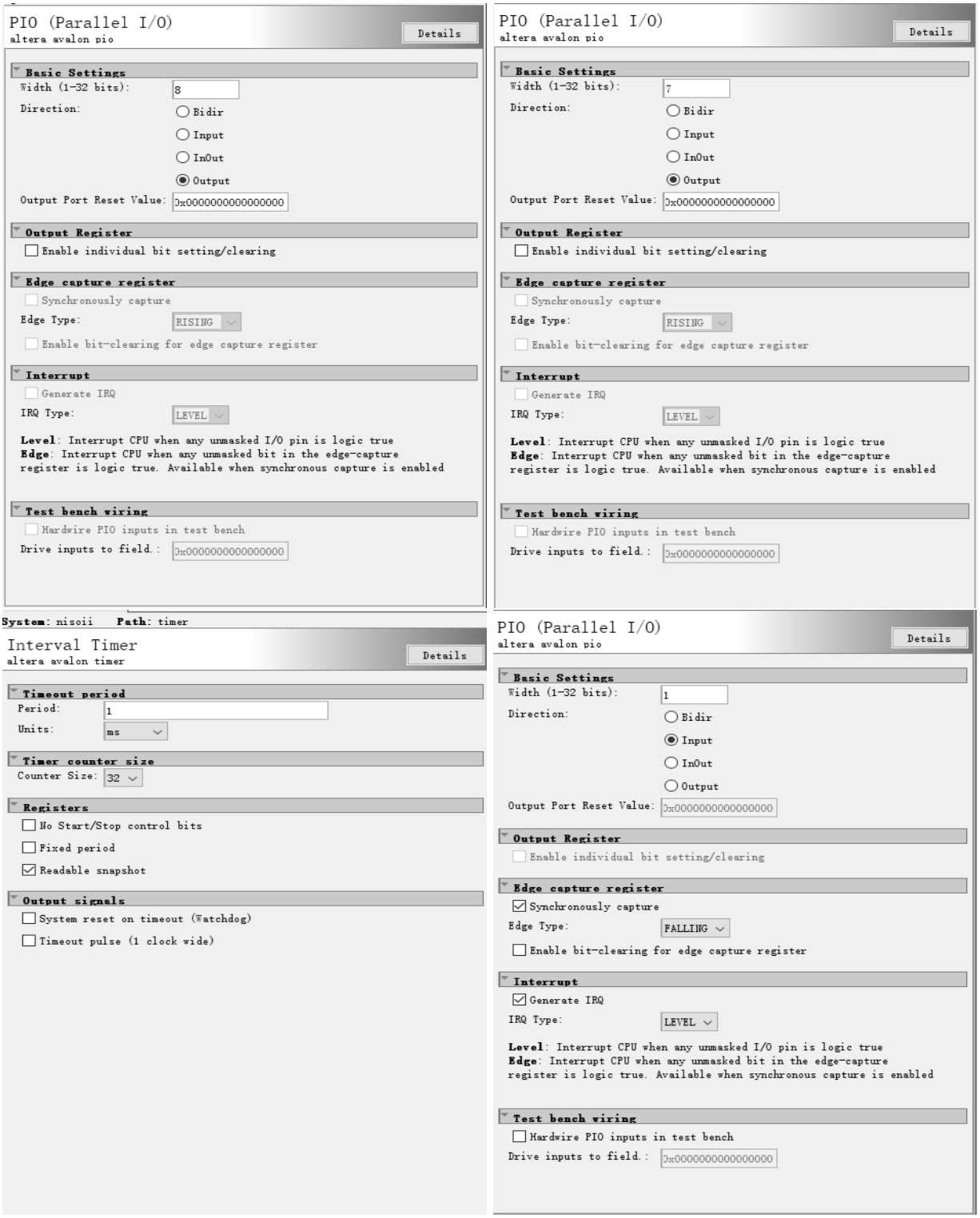

图 3: 各个连接部件参数

(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。

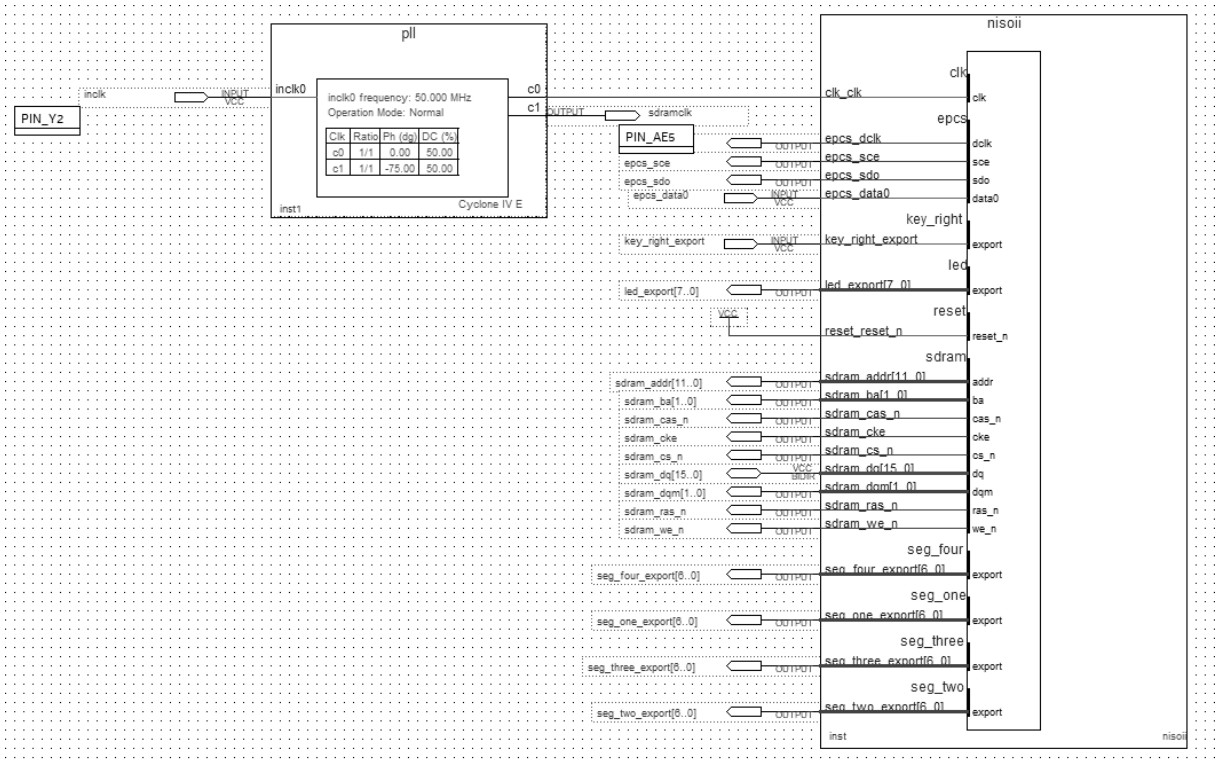

图 4: bdf 文件绘制

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
LEDG[1]	PIN_E22	LED Green[1]	2.5V
LEDG[2]	PIN_E25	LED Green[2]	2.5V
LEDG[3]	PIN_E24	LED Green[3]	2.5V
LEDG[4]	PIN_H21	LED Green[4]	2.5V
LEDG[5]	PIN_G20	LED Green[5]	2.5V
LEDG[6]	PIN_G22	LED Green[6]	2.5V
LEDG[7]	PIN_G21	LED Green[7]	2.5V
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。

信号名	FPGA 引脚号	说明	I/O 标准
HEX0[0]	PIN_G18	Seven Segment Digit 0[0]	2.5V
HEX0[1]	PIN_F22	Seven Segment Digit 0[1]	2.5V
HEX0[2]	PIN_E17	Seven Segment Digit 0[2]	2.5V
HEX0[3]	PIN_L26	Seven Segment Digit 0[3]	取决于 JP7
HEX0[4]	PIN_L25	Seven Segment Digit 0[4]	取决于 JP7
HEX0[5]	PIN_J22	Seven Segment Digit 0[5]	取决于 JP7
HEX0[6]	PIN_H22	Seven Segment Digit 0[6]	取决于 JP7
HEX1[0]	PIN_M24	Seven Segment Digit 1[0]	取决于 JP7
HEX1[1]	PIN_Y22	Seven Segment Digit 1[1]	取决于 JP7
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]	取决于 JP7
HEX1[3]	PIN_W22	Seven Segment Digit 1[3]	取决于 JP7
HEX1[4]	PIN_W25	Seven Segment Digit 1[4]	取决于 JP7

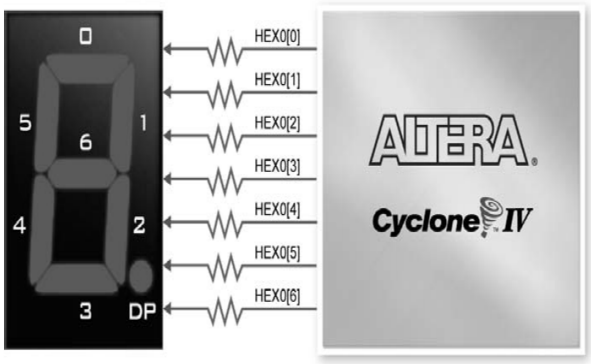

图 6: (1) 数码管管脚分配 (截取部分)；(2) 数码管与 FPGA 连接

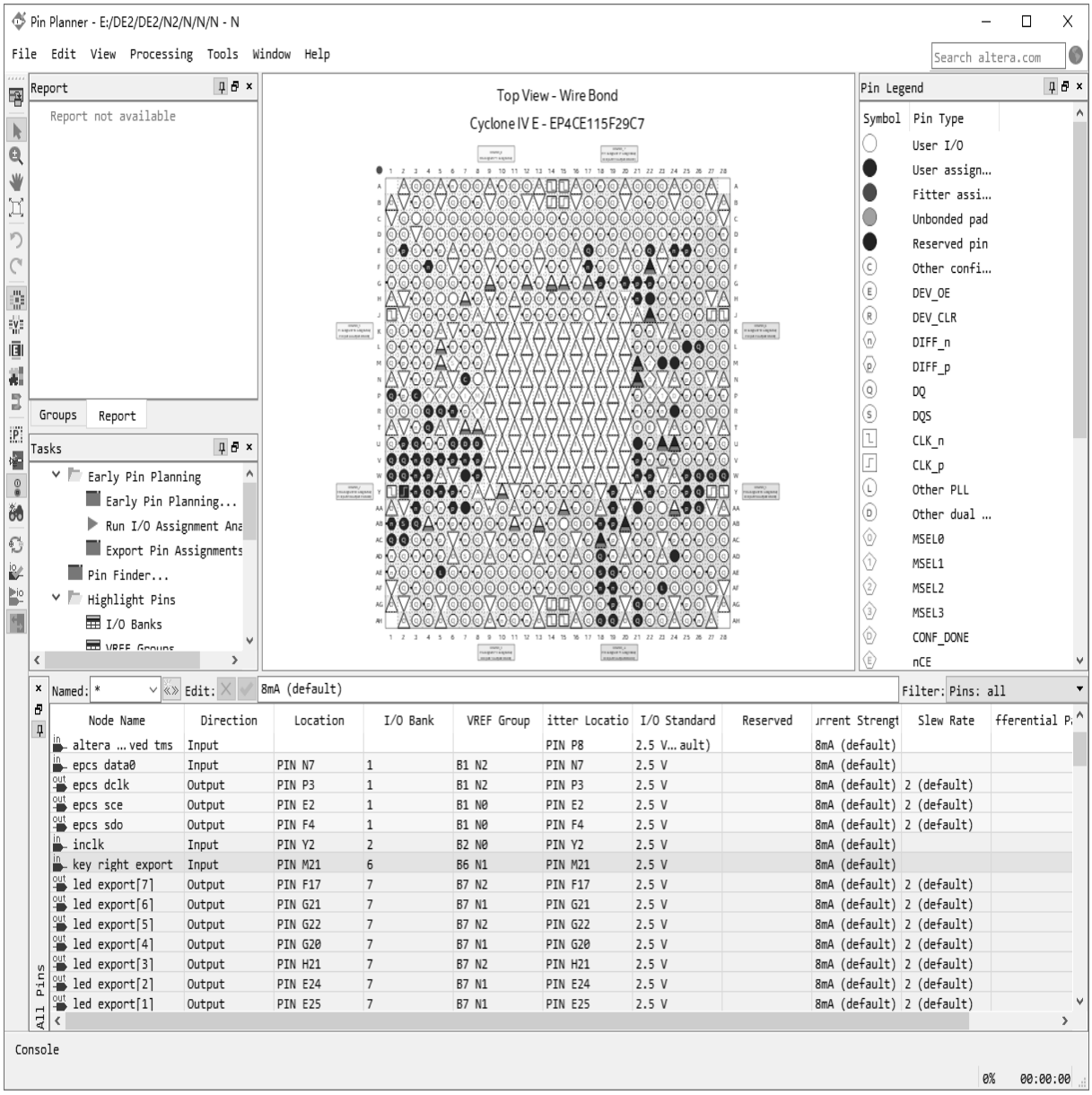

图 7: 最终的引脚分配

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>
#include "system.h"
#include "sys/alt_irq.h"
#include "sys/alt_alarm.h"
#include "altera_avalon_pio_regs.h"
#include "altera_avalon_timer_regs.h"
#include "unistd.h"
#include "alt_types.h"

typedef struct
{
                unsigned long int DATA;
                unsigned long int DIRECTION;
                unsigned long int INTERRUPT_MASK;
                unsigned long int EDGE_CAPTURE;
}PIO_STR;

typedef struct          //定时器寄存器
{
                unsigned long int STATUS;
                unsigned long int CONTROL;
                unsigned long int PERIOD_L;
                unsigned long int PERIOD_H;
                unsigned long int SNAP_L;
                unsigned long int SNAP_H;
}TIMER_STR;

#define SEG_ONE             ((PIO_STR*)SEG_ONE_BASE)
#define SEG_TWO             ((PIO_STR*)SEG_TWO_BASE)
#define SEG_THREE           ((PIO_STR*)SEG_THREE_BASE)
#define SEG_FOUR            ((PIO_STR*)SEG_FOUR_BASE)
#define TIMER               ((TIMER_STR*)TIMER_BASE)    //指向定时器结构体
#define KEY_RIGHT           ((PIO_STR*)KEY_RIGHT_BASE)
```

```
34
35 unsigned char table[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
36 unsigned char second = 0;
37 unsigned char minute = 0;
38 unsigned char hour = 0;
39
40 alt_u8 led8 = 0x02;      // 0000 0010
41 alt_u8 dir = 0;          // 0000 0000
42 volatile int i;
43
44 void ISR_Timer(void *context,alt_u32 id)  //定时器中断服务函数
45 {
46                 second++;
47                 TIMER->STATUS &= 0x00;  //清中断标志位
48 }
49 //显示函数
50 void display(unsigned char hour,unsigned char min,unsigned char sec)
51 {
52                 unsigned char ge,shi;
53                 ge = sec%10;
54                 shi = sec/10;
55
56                 SEG_ONE -> DATA = 0xff;
57                 SEG_ONE -> DATA = table[ge];
58                 usleep(1000);
59
60                 SEG_TWO -> DATA = 0xff;
61                 SEG_TWO -> DATA = table[shi];
62                 usleep(1000);
63
64                 ge = min%10;
65                 shi = min/10;
66
67                 SEG_THREE -> DATA = 0xff;
68                 SEG_THREE -> DATA = table[ge];
69                 usleep(1000);
70
71                 SEG_FOUR -> DATA = 0xff;
72                 SEG_FOUR -> DATA = table[shi];
73                 usleep(1000);
74
75 }
76
```

```
77  int main(void)  //主函数
78  {
79                  //注册中断
80                  alt_irq_register(TIMER_IRQ,(void*)TIMER_BASE,ISR_Timer);
81                  //初始化中断周期,周期为一秒
82                  TIMER -> PERIOD_L = 50000000&0xffff;
83                  TIMER -> PERIOD_H = 50000000>>16;
84                  TIMER -> CONTROL = 0x07;  //启动定时器并开中断
85
86  while(1)
87      {
88  //seg
89                  if(second > 59)
90                          {
91                                  second = 0;
92                                  minute++;
93                          }
94                  if(minute > 59)
95                          {
96                                  minute = 0;
97                                  hour++;
98                          }
99                  if(hour>23)
100                         {
101                                 hour = 0;
102                         }
103
104                 display(hour,minute,second);
105
106                 /*if(KEY_LEFT -> DATA == 0)  //按键控制
107                 {
108                                 usleep(20000);
109                                 hour++;
110                 }*/
111                 if(KEY_RIGHT -> DATA == 0)
112                 {
113                                 usleep(20000);
114                                 minute++;
115         }
116
117 //led
118                 //led=1000 0000/0000 0001时候,这个if语句都会执行
119                 if (led8 & 0x81)
```

```
120                 {
121                         dir = (dir ^ 0x01); //1
122             }
123             if (dir) //1
124             {
125                         led8 = led8 >> 1;    //LED右移动显示
126                  }
127                 else
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;
135                     while (i<250000)   //延时
136                         i++;
137     }
138                 return 0;
139 }
```