**数字系统（课程）设计**

**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **数字系统设计实验** |
| **学生姓名：** | **蒲 尧** |
| **学生学号：** | **201630258438** |
| **学生专业：** | **信息工程** |
| **开课学期：** | **2018-2019第一学期** |
| **实验成绩：** |  |

**电子信息学院**

**2016年5月**

**电子与信息学院本科教学实验室**

**学生实验安全操作规则**

为培养学生严谨的学习作风，营造安全的实验实践学习氛围，确保学生人身和仪器设备安全，顺利完成实验任务，特制定以下规则。

1、不得赤脚或穿拖鞋进入实验室，不得将食物带入实验室，不准随地丢弃废纸、废弃物，保持实验室清洁。

2、实验中不得触摸裸露的接线柱、接线片、导线，不得用表棒、镊子、剪刀等工具插入电源插座。严禁带电接线、拆线或改接线路。

3、电烙铁在通电而不用时，应始终置于烙铁架内，严禁将其随意摆在桌边或地上；留意烙铁头避开导线及附近的仪器设备和书籍用品；人体不可触碰烙铁头，以防烫伤或漏电事故。中途离开实验室或实验完毕，要及时断电，以免发生火灾事故。

4、实验接线完毕，要仔细复查，确认无误后方可接通电源。

5、不要动用与本次实验无关的设备、仪器、工具和原材料。

6、实验结束后，实验装置、仪器仪表及电烙铁要断电，将实验用过的有关的仪器、工具、导线及电缆整理好，放回原处；并将座椅推至实验台下方可离场。

7、遇到事故应立即断电，并及时向实验指导教师报告。

8、严守纪律，遵守实验室规章和安全制度，严格执行操作规则。

9、提高防火防盗意识，做好防火防盗工作。

电子与信息学院实验中心

二〇一六年七月

**本人已阅读上述《学生实验安全操作规则》，知悉规则条款，并承诺认真执行。**

承 诺 人： 蒲 尧

学 院： 电子与信息学院

专业/班级： 信息工程3班

时 间： 2018.10.17

**目 录**

[实验一 Quartus软件的使用；VHDL程序结构的熟悉 3](#_Toc531175403)

[实验二 基本电路模块设计（组合、时序） 7](#_Toc531175404)

[实验三 基于状态机的交通灯控制 13](#_Toc531175405)

[实验四 按键控制的状态机设计 27](#_Toc531175406)

# Quartus软件的使用；VHDL程序结构的熟悉

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | 31号楼 | 312房 | **实验台号：** | 23 |
| **实验日期与时间：** | 2018.10.17 | | **评 分：** |  |
| **预习检查纪录：** |  | | **批改教师：** |  |

1. 实验要求：

1. 熟悉 Quartus 软件的使用

2. 熟悉 VHDL 程序结构

1. 实验内容：

按新教材实例（P120 页开始，详见附件资料）操作，掌握建立工程、设计输入、编译、时序仿真等开发流程；至有仿真结果为止。要理解 VHDL 代码意思，将软件和硬件结合起来，掌握开发流程。特别提醒：工程名一定要同顶层文件名（总电路图）一致；实体名一定要同当前待编译文件名一致；当前文件做编译之前一定要先设为顶层文件（点击 Project 中 Set as Top-Level Entity）。

1. 实验设计原理；

QuartusⅡ的开发流程：设计输入->综合->布线布局->时序分析->仿真->编程配置，在 Quartus II 中综合、布线布局、时序分析都包含在编译中，也就是在点击 Start Complication 后，软件会自动完成这三部分的功能。设计输入一般有文本输入、图形输入、网表输入等几种方法，仿真之前需要编辑好输入信号激励文件，编程配置则需要根据开发板的说明书来对系统的输入输出管脚进行配置。

此外，Quartus II 软件为设计流程的每个阶段提供 Quartus II 图形用户界面、EDA 工具界面以及命令行界面。可以在整个流程中只使用这些界面中的一个，也可以在设计流程的不同阶段使用不同界面。

四, 实验过程记录（流程图或者实验逻辑思路过程）

**流程图：**

**源代码：**

delay.vhd文件：

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** delay **is**

**port(**clk**,**rst**:in** std\_logic**;**

pul**:out** std\_logic**);**

**end** delay**;**

**architecture** behav **of** delay **is**

**signal** cnt**:**std\_logic\_vector**(**23 **downto** 0**);**

**begin**

**process(**clk**,**rst**)**

**begin**

**if(**rst**=**'0'**)then**

cnt**<=**"000000000000000000000000"**; --当按下键时，计数器从0开始**

**elsif(**clk'**event** and clk**=**'1'**)then**

**if(**cnt**=**"100110001001011001111110"**)then --每隔不到20ms产生一个脉冲**

cnt**<=**"100110001001011001111110"**;**pul**<=**'0'**;**

**else** cnt**<=**cnt**+**'1'**;**pul**<=**'1'**;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** behav**;**

light.vhd文件：

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** light **is**

**port(**pul**,**rst**:in** std\_logic**;**

led**:out** std\_logic\_vector**(**5 **downto** 0**));**

**end** light**;**

**architecture** behav **of** light **is**

**signal** i**:**std\_logic\_vector**(**2 **downto** 0**);**

**begin**

**process(**pul**,**rst**)**

**begin**

**if(**rst**=**'0'**)then**

led**<=**"000000"**;**i**<=**"000"**;**

**elsif(**pul'**event** and pul**=**'1'**)then**

**if(**i**=**5**)then**

i**<=**"000"**;**

**else**

i**<=**i**+**'1'**;**

**end** **if;**

**case** i **is**

**when** "000"**=>**led**<=**"111110"**;**

**when** "001"**=>**led**<=**"111101"**;**

**when** "010"**=>**led**<=**"111011"**;**

**when** "011"**=>**led**<=**"110111"**;**

**when** "100"**=>**led**<=**"110111"**;**

**when** **others=>**led**<=**"011111"**;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** behav**;**

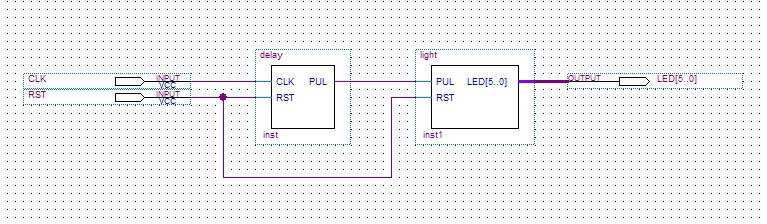
五，实验结果和分析

能够根据书中代码正确编写vhdl代码生成vhd文件，生成延时和控制亮灯两个模块，并进行电路连接最终实现正确仿真，实验电路图和仿真图如下。

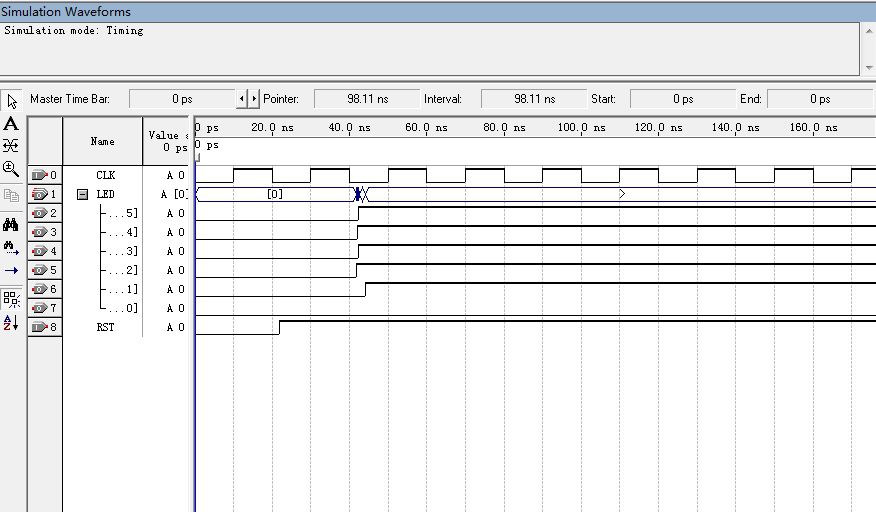
**（实验报告作品相片粘贴页）**

（总电路图截图、仿真结果或者实验平台运行效果照片）

**连接好的整个系统图：**



**时序仿真图：**



# 基本电路模块设计（组合、时序）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | 31号楼 | 312房； | **实验台号：** | 23 |
| **实验日期与时间：** | 2018.10.31 | | **评 分：** |  |
| **预习检查纪录：** |  | | **批改教师：** |  |

报告内容：

一，实验要求：

1、实现3-8 译码器

2、完成分频器实验

1. 实验内容：

1、3-8 译码器

设计要求：

1)3个译码输入端 A、B、C ，分别对应开发板上表示为 ON DIP 的 2,3,4

2) EN为使能端（低电平有效），对应开发板上表示为 ON DIP 的 1 管脚

3) Y为译码输出，8 位位矢量类型。D3 至 D10

4)输入采用电平开关，译码输出采用 LED 指示灯显示

5)管脚对应见附件：信号分配表

6)译码表在下面

2、分频器实验

设计要求：

1) 将 EDA 板上的系统时钟 50MHz 分频为 1Hz 的时钟信号

2) 占空比为 50%

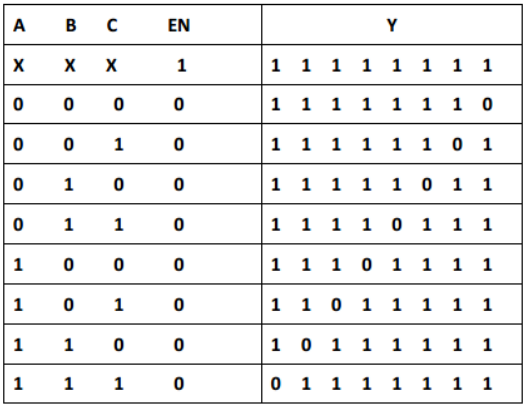
3) 利用流水灯点亮程序，在 EDA 板上观察效果

4) 实体命名为 clkdiv\_(班级号)\_(班级序号)

注：由于开发板上没有 rst\_n 按钮，同学们将 rst\_n 直接接高电平。

1. 实验设计原理；

1、3-8 译码器



2、分频器实验：计数与分频

四, 实验过程记录（流程图或者实验逻辑思路过程）

五，实验结果和分析

实验一成功编写vhdl代码生成模块，并编译、下载成功。能够在使能信号为1（即不按下时），根据abc三位输入，点亮相应的灯（即译码成功）。

实验二成功编写两个vhdl代码并生成模块，并编译、下载成功。能够将50MHz时钟信号分频成1Hz并以流水灯形式输出，实验成功。

六，附录（打印源代码页）

**--decoder.vhd**

**library** IEEE**;**

**use** IEEE**.**std\_logic\_1164**.all;**

**entity** decoder **is**

**port(**en, a**,**b**,**c**:in** std\_logic**;**

y**:out** std\_logic\_vector**(**7 **downto** 0**)**

**);**

**end** **entity;**

**architecture** behav **of** decoder **is**

**signal** s**:**std\_logic\_vector**(**2 **downto** 0**);**

**begin**

s**<=**a**&**b**&**c**;**

**process(**en**,**s**)**

**begin**

y**<=(others=>**'1'**);**

**if** en**=**'1' **then**

**case** s **is**

**when** "000"**=>**y**<=**"11111110"**;** --0

**when** "001"**=>**y**<=**"11111101"**;** --1

**when** "010"**=>**y**<=**"11111011"**;** --2

**when** "011"**=>**y**<=**"11110111"**;** --3

**when** "100"**=>**y**<=**"11101111"**;** --4

**when** "101"**=>**y**<=**"11011111"**;** --5

**when** "110"**=>**y**<=**"10111111"**;** --6

**when** "111"**=>**y**<=**"01111111"**;** --7

**when** **others=>null;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** **architecture;**

**--****divider.vhd**

**library** ieee**;**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_arith**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** divider **is**

**generic(**n**:**integer**:=**50000000**);** --N times divide

**port(**clkin**:in** std\_logic**;**

clkout**:out** std\_logic**);**

**end** **entity;**

**architecture** behav **of** divider **is**

**signal** cnt**:**integer **range** 0 **to** n**-**1**;**

**begin**

**process(**clkin**)** --count process

**begin**

**if(**clkin'**event** and clkin**=**'1'**)then**

**if(**cnt**<**n**-**1**)then**

cnt**<=**cnt**+**1**;**

**else**

cnt**<=**0**;**

**end** **if;**

**end** **if;**

**end** **process;**

**process(**cnt**)**

**begin**

**if(**cnt**<(**n**/**2**))then**

clkout**<=**'1'**;**

**else**

clkout**<=**'0'**;**

**end** **if;**

**end** **process;**

**end** behav**;**

**--lighten.vhd**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_arith**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** lighten **is**

**port(**clk**,**rst**:in** std\_logic**;**

led\_out**:out** std\_logic\_vector**(**5 **downto** 0**));**

**end** **entity;**

**architecture** behav **of** lighten **is**

**signal** light**:**std\_logic\_vector**(**5 **downto** 0**);**

**begin**

**process(**clk**,**rst**)**

**begin**

**if(**rst**=**'0'**)then**

light**<=**"00000010"**;**

**elsif(**clk'**event** and clk**=**'1'**)then**

**if(**light**=**"000000"**)then**

light**<=**"000001"**;**

**else**

light**<=**light**(**4 **downto** 0**)&** light**(**5**);**

**end** **if;**

**end** **if;**

**end** **process;**

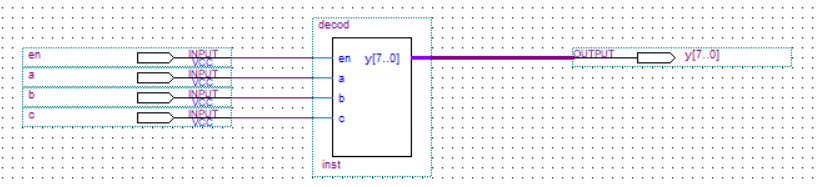
led\_out**<=**light**;**

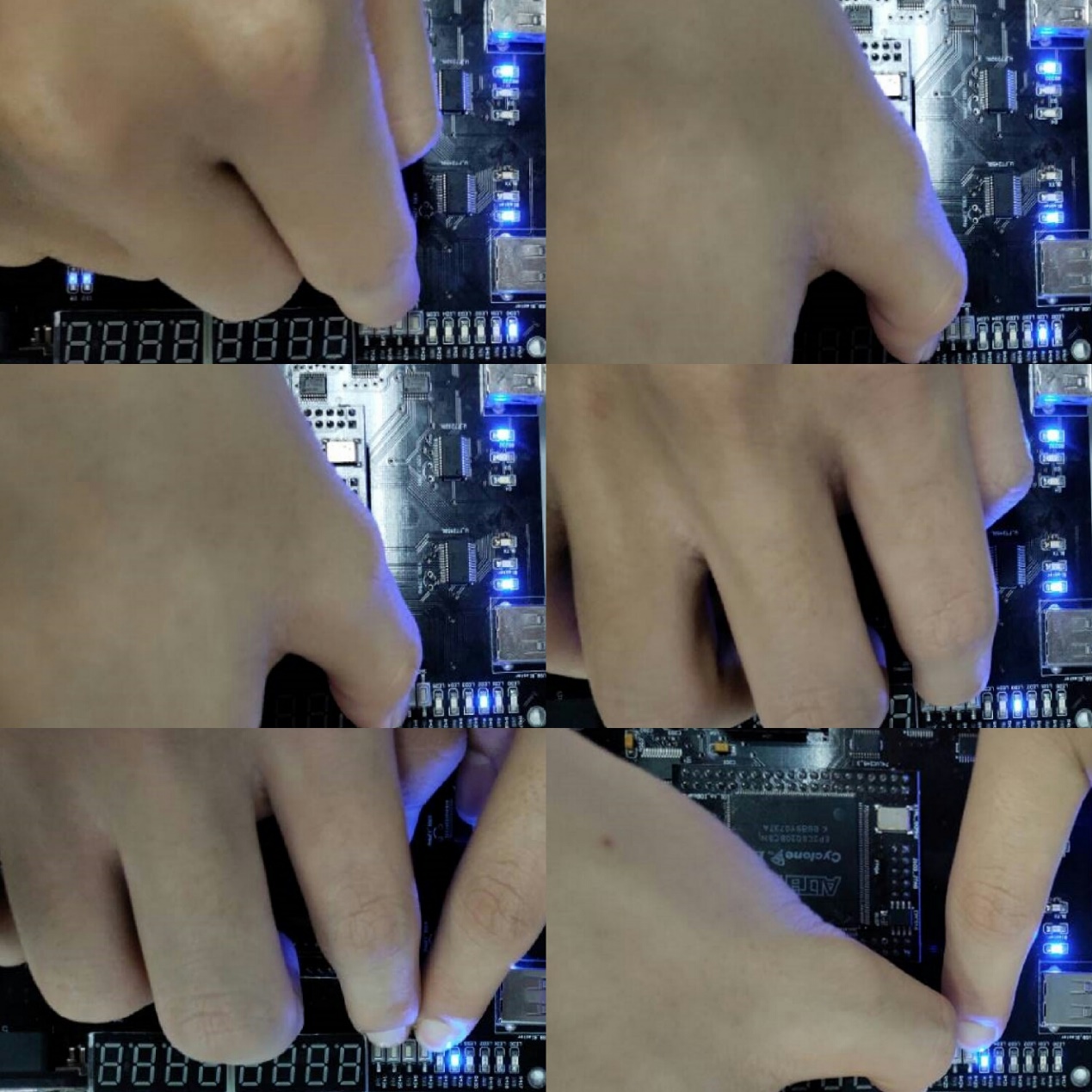
**end** behav**;**

**（实验报告作品相片粘贴页）**

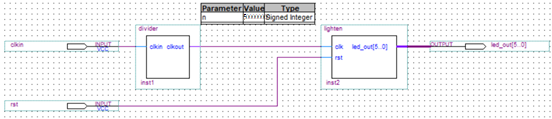
（总电路图截图、仿真结果或者实验平台运行效果照片）

3-8decoder

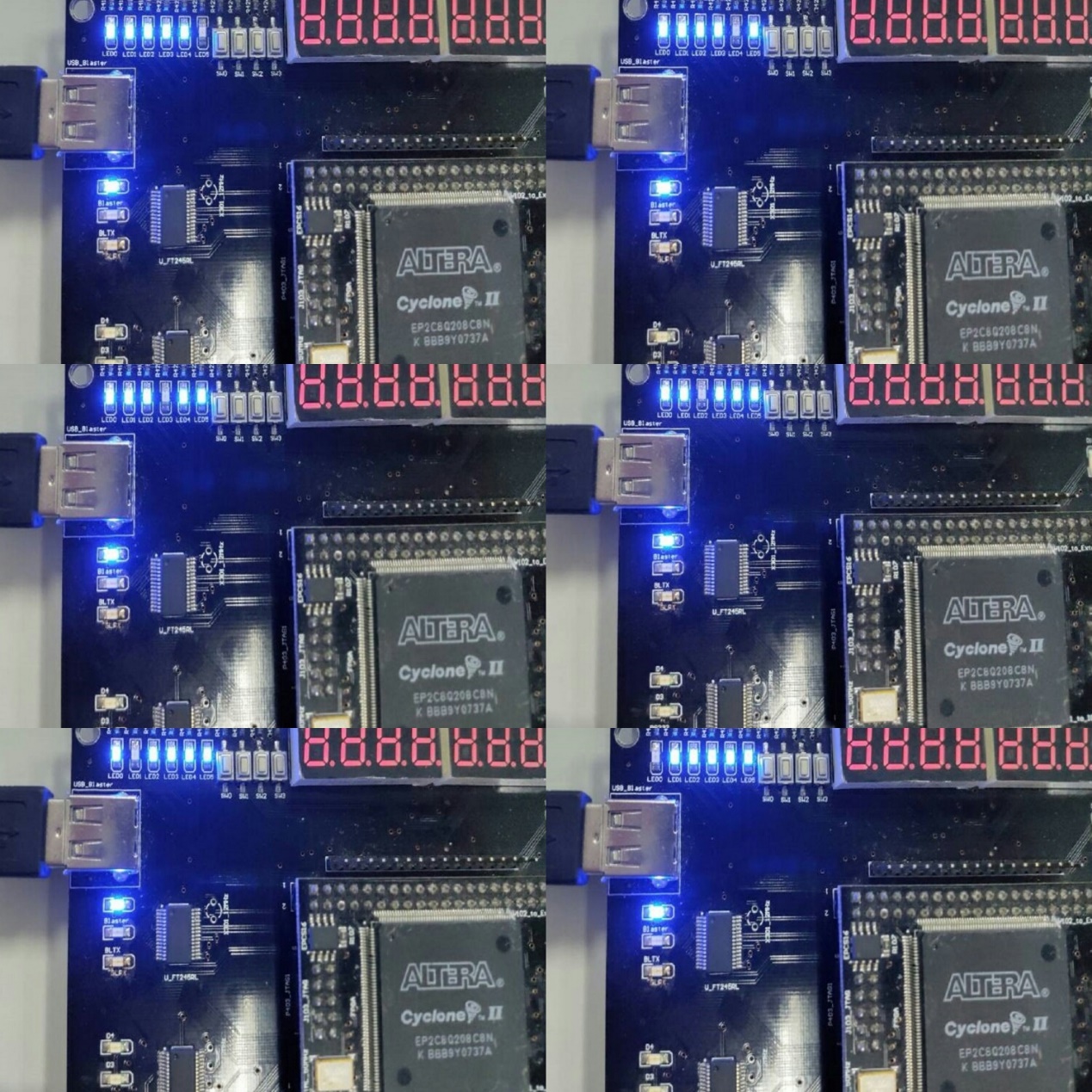




Divider



流水灯1~8



# 实验三 基于状态机的交通灯控制

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | 31号 楼 | 312 房； | **实验台号：** | 23 |
| **实验日期与时间：** | 2018.11.14 | | **评 分：** |  |
| **预习检查纪录：** |  | | **批改教师：** |  |

报告内容：

1. 实验要求：

运用状态机设计红黄绿交通灯控制电路系统

1. 实验内容：

1、 开发板上三个 led 等分别代表公路上红黄绿三种颜色交通灯。

2、 交通灯状态机初始状态为红灯，交通灯工作过程依次是红 绿 黄 红。

3、 为了方便观察，本次实验要求红灯的显示时间为 9s，绿灯显示时间为 6s ，黄灯显示时间为 3s ，时间需要倒计时，在数码管上显示。编程之前要求同学们先画好 ASM 图。

4、 1Hz 分频模块请采用第二次实验中的内容，7 段码显示模块请参考书本相关内容。

5、 第三次实验课用到 EP2C8Q208C8 通过 74HC595 驱动数码管，有两种方法写该模块代码：方法 1，用 VHDL 语言写，（自己写 VHDL 代码有加实验分）。方法 2，可调用 verilog 数码管驱动模块，该模块在附件“seg.zip”中。和其它VHDL编写的模块可以混搭在一个电路图中使用。EP2C8Q208C8的SCTP，SHCP，SER＿DATA 数码管信号线通过两块 74HC595 集成块，再驱动数码管。

6、 芯片型号：cyclone:EP2C8Q208C8,开发板所有资料都在“新板”附件中，其中管脚配置在实验要求中是不对的，以“新板”附件中为准。

1. 实验设计原理；

根据实验内容，该交通灯的控制可以作为一个有限状态机来设计。输出信号包括red、green、yellow，分别表示红灯、绿灯、黄灯。red、green、yellow均为std\_logic类型，高电平有效。选择单进程的设计方式，在一个进程中描述状态寄存器、次态以及输出逻辑。进程说明部分定义整型变量cnt实现计数功能。

四, 实验过程记录（流程图或者实验逻辑思路过程）

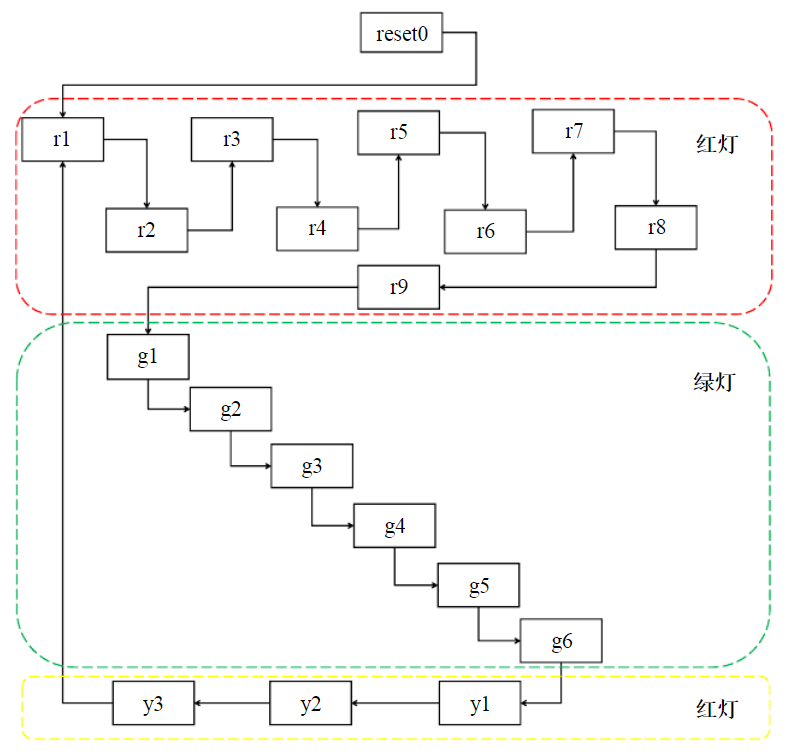
1.启动软件，建立新的工程

2.设计输入，分别输入分频器模块divider.vhdl、交通灯模块traffic\_light.vhdl、位数转换模块bitstransfer.vhdl、数码管模块seg.vhdl。

3.链接好并重命名后，保存文件并编译。

4.时序仿真，查看仿真结果。

5.安装驱动，将程序下载到开发板芯片中，直接在芯片上观察实验效果。



五，实验结果和分析

把程序下载到开发板后，可见代表红灯的LED灯先亮，同时数码管从9开始倒计时。倒计时结束后，代表绿灯的LED灯亮，数码管从6开始倒计时。然后代表黄灯的LED灯亮，数码管从3开始倒计时。结束后重新从红灯开始，如此循环。与实验要求一致，可见代码基本正确。

六，附录（打印源代码页）

**第一个简单工程代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_arith**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** led **is**

**port(**led\_out **:** **out** std\_logic\_vector**(**7 **downto** 0**);**

clk **:** **in** std\_logic**;**

rst\_n **:** **in** std\_logic

**);**

**end** led**;**

**architecture** behavior **of** led **is**

**signal** light **:** std\_logic\_vector **(**7 **downto** 0**);**

**begin**

**process(**clk**,** rst\_n**)**

**begin**

**if(**rst\_n**=**'0'**)then**

light **<=** "00000010"**;**

**elsif(**clk' **event** and clk**=**'1'**)then**

**if(**light **=** "00000000" **)** **then**

light **<=** "00000001"**;**

**else**

**if(**light **=** "10000000" **)** **then**

light **<=** "00000001"**;**

**else**

light **<=** light**(**6 **downto** 0**)&**'0'**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

led\_out **<=** light**;**

**end** behavior**;**

**第二个工程代码**

--(1) divider

**LIBRARY** IEEE**;**

**USE** IEEE**.**std\_logic\_1164**.ALL;**

**USE** IEEE**.**std\_logic\_unsigned**.all;**

**ENTITY** divider **is**

**GENERIC(**n**:**integer**:=**50000000**);**

**PORT(**clk**:IN** std\_logic**;**

Y**:OUT** std\_logic**);**

**END** **ENTITY;**

**ARCHITECTURE** behav **OF** divider **IS**

**SIGNAL** count**:**integer **RANGE** n**-**1 **DOWNTO** 0**:=**n**-**1**;**

**BEGIN**

**PROCESS(**clk**)**

**BEGIN**

**IF(rising\_edge(**clk**))** **then** count**<=**count**-**1**;**

**if(**count**>=**n**/**2**)** **then** Y**<=**'0'**;**

**else** Y**<=**'1'**;**

**end** **if;**

**if(**count**<=**0**)** **then** count**<=**n**-**1**;**

**end** **if;**

**END** **IF;**

**END** **PROCESS;**

**END** behav**;**

--(2) traffic\_light

**LIBRARY** IEEE**;**

**USE** IEEE**.**std\_logic\_1164**.ALL;**

**USE** IEEE**.**std\_logic\_unsigned**.all;**

**ENTITY** traffic\_light **is**

**PORT(**clk**:IN** std\_logic**;**

rst\_n**:IN** std\_logic**;**

red\_light**:OUT** std\_logic**;**

green\_light**:OUT** std\_logic**;**

yellow\_light**:OUT** std\_logic**;**

holdtime**:OUT** integer **range** 0 **TO** 9**);**

**END** **ENTITY;**

**ARCHITECTURE** Behav **OF** traffic\_light **IS**

**TYPE** state\_type **IS(**reset0**,**r1**,**r2**,**r3**,**r4**,**r5**,**r6**,**r7**,**r8**,**r9**,**g1**,**g2**,**g3**,**g4**,**g5**,**g6**,**y1**,**y2**,**y3**);**

**SIGNAL** present\_state**,**next\_state**:**state\_type**;**

**BEGIN**

--seq process:create state

seq**:PROCESS(**clk**,**rst\_n**)**

**BEGIN**

**IF(**rst\_n**=**'0'**)** **THEN** present\_state**<=**reset0**;**

**ELSIF(rising\_edge(**clk**))** **THEN** present\_state**<=**next\_state**;**

**END** **IF;**

**END** **PROCESS;**

--com process:create next state and output logic com:

**PROCESS(**next\_state**)**

**BEGIN**

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**0**;**

**CASE** present\_state **IS**

**WHEN** reset0**=>**

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**0**;**

next\_state**<=**r1**;**

**WHEN** r1**=>**--red light reset time=9

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**9**;**

next\_state**<=**r2**;**

**WHEN** r2**=>**--red light reset time=8

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**8**;**

next\_state**<=**r3**;**

**WHEN** r3**=>**-- red light reset time =7

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**7**;**

next\_state**<=**r4**;**

**WHEN** r4**=>**-- red light reset time =6

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**6**;**

next\_state**<=**r5**;**

**WHEN** r5**=>**-- red light reset time =5

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**5**;**

next\_state**<=**r6**;**

**WHEN** r6**=>**-- red light reset time =4

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**4**;**

next\_state**<=**r7**;**

**WHEN** r7**=>**-- red light reset time =3

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**3**;**

next\_state**<=**r8**;**

**WHEN** r8**=>**-- red light reset time =2

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**2**;**

next\_state**<=**r9**;**

**WHEN** r9**=>**-- red light reset time =1

red\_light**<=**'0'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**1**;**

next\_state**<=**g1**;**

**WHEN** g1**=>**--green light reset time =6

red\_light**<=**'1'**;**

green\_light**<=**'0'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**6**;**

next\_state**<=**g2**;**

**WHEN** g2**=>**-- green light reset time =5

red\_light**<=**'1'**;**

green\_light**<=**'0'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**5**;**

next\_state**<=**g3**;**

**WHEN** g3**=>**-- green light reset time =4

red\_light**<=**'1'**;**

green\_light**<=**'0'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**4**;**

next\_state**<=**g4**;**

**WHEN** g4**=>**-- green light reset time =3

red\_light**<=**'1'**;**

green\_light**<=**'0'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**3**;**

next\_state**<=**g5**;**

**WHEN** g5**=>**-- green light reset time =2

red\_light**<=**'1'**;**

green\_light**<=**'0'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**2**;**

next\_state**<=**g6**;**

**WHEN** g6**=>**-- green light reset time =1

red\_light**<=**'1'**;**

green\_light**<=**'0'**;**

yellow\_light**<=**'1'**;**

holdtime**<=**1**;**

next\_state**<=**y1**;**

**WHEN** y1**=>**-- yellow light reset time =3

red\_light**<=**'1'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'0'**;**

holdtime**<=**3**;**

next\_state**<=**y2**;**

**WHEN** y2**=>**-- yellow light reset time =2

red\_light**<=**'1'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'0'**;**

holdtime**<=**2**;**

next\_state**<=**y3**;**

**WHEN** y3**=>**-- yellow light reset time =1

red\_light**<=**'1'**;**

green\_light**<=**'1'**;**

yellow\_light**<=**'0'**;**

holdtime**<=**1**;**

next\_state**<=**r1**;**

**END** **CASE;**

**END** **PROCESS;**

**END** Behav**;**

--(3)将4位二进制数转为16位二进制数描述

**LIBRARY** IEEE**;**

**USE** IEEE**.**std\_logic\_1164**.ALL;**

**USE** IEEE**.**std\_logic\_unsigned**.all;**

**ENTITY** bitstransfer **is**

**PORT(**clk**:IN** std\_logic**;**

input4bit**:IN** std\_logic\_vector**(**3 **DOWNTO** 0**);**

output16bit**:OUT** std\_logic\_vector**(**15 **DOWNTO** 0**));**

**END** **ENTITY;**

**ARCHITECTURE** Behav **OF** bitstransfer **IS**

**BEGIN**

output16bit**<=**"ZZZZZZZZZZZZZZZZ"**;**

**PROCESS(**clk**,**input4bit**)**

**BEGIN**

**IF(rising\_edge(**clk**))THEN**

output16bit**<=**"000000000000"**&**input4bit**;**

**END** **IF;**

**END** **PROCESS;**

**END** behav**;**

--seg.v

module seg**(**

clk**,**rst\_n**,**en**,**

idis\_data**,**

ds\_stcp**,**ds\_shcp**,**ds\_data

**);**

input clk**;** **//**25M**??????**

input rst\_n**;** **//??????????**

input en**;**

input **[**15**:**0**]** idis\_data **;**

output ds\_stcp**;** **//**74HC595**????????????????????????**

output ds\_shcp**;** **//**74HC595**?????????????????????**

output ds\_data**;** **//**74HC595**???????**

**//?????** 0**~**F **??????**

**parameter** SEG\_NUM0 **=** 8'hc0**,**

SEG\_NUM1 **=** 8'hf9**,**

SEG\_NUM2 **=** 8'ha4**,**

SEG\_NUM3 **=** 8'hb0**,**

SEG\_NUM4 **=** 8'h99**,**

SEG\_NUM5 **=** 8'h92**,**

SEG\_NUM6 **=** 8'h82**,**

SEG\_NUM7 **=** 8'hF8**,**

SEG\_NUM8 **=** 8'h80**,**

SEG\_NUM9 **=** 8'h90**,**

SEG\_NUMA **=** 8'h88**,**

SEG\_NUMB **=** 8'h83**,**

SEG\_NUMC **=** 8'hc6**,**

SEG\_NUMD **=** 8'ha1**,**

SEG\_NUME **=** 8'h86**,**

SEG\_NUMF **=** 8'h8e**;**

**//?????** 0**~**7**????**

**parameter** SEG\_WE0 **=** 8'b1111\_1110**,**

SEG\_WE1 **=** 8'b1111\_1101**,**

SEG\_WE2 **=** 8'b1111\_1011**,**

SEG\_WE3 **=** 8'b1111\_0111**;**

**//** SEG\_WE4 **=** 8'b1110\_1111**,**

**//** SEG\_WE5 **=** 8'b1101\_1111**,**

**//** SEG\_WE6 **=** 8'b1011\_1111**,**

**//** SEG\_WE7 **=** 8'b0111\_1111**;**

wire en**;**

reg clk\_div\_2**;**

reg clk1**;**

always@**(**en**)** **//use** enable siganl **to** break the module

**begin**

**if** **(**en**==**1**)**

clk1**<=**clk**;**

**else** clk1**<=**clk1**;**

**end**

always@**(**posedge clk1 or negedge rst\_n**)**

**if(!**rst\_n**)**

clk\_div\_2**<=**1'b0**;**

**else**

clk\_div\_2**<=~**clk\_div\_2**;**

**//**-------------------------------------------------

**//**-------------------------------------------------

**//??????????**

reg**[**3**:**0**]** seg\_num**;** **//??????**

reg**[**7**:**0**]** seg\_duan**;** **//**7**???????????????**8**??**

reg**[**7**:**0**]** seg\_wei**;** **//**7**????????**

reg**[**7**:**0**]** cnt\_4**;** **//?????**

**//?????**

always @**(**posedge clk\_div\_2 or negedge rst\_n**)**

**if(!**rst\_n**)** cnt\_4 **<=** 8'd0**;**

**else** cnt\_4 **<=** cnt\_4**+**1'b1**;**

**//????**

always @**(**posedge clk\_div\_2 or negedge rst\_n**)**

**if(!**rst\_n**)** seg\_num **<=** 8'h00**;**

**else**

**case(**cnt\_4**[**7**:**6**])**

2'b00**:** seg\_num **<=** idis\_data**[**3**:**0**];**

2'b01**:** seg\_num **<=** idis\_data**[**7**:**4**];**

2'b10**:** seg\_num **<=** idis\_data**[**11**:**8**];**

2'b11**:** seg\_num **<=** idis\_data**[**15**:**12**];**

**default:** **;**

endcase

**//??????**

reg flag**;**

always @**(**posedge clk\_div\_2 or negedge rst\_n**)**

**if(!**rst\_n**)** **begin** seg\_duan **<=** 8'hff**;**

**//** flag**<=**1'b0**;**

**end**

**//else** **if(**flag**)** **begin** seg\_duan**<=**8'hff**;**

**//** flag**<=~**flag**;**

**//** **end**

**else**

**case(**seg\_num**)**

4'h0**:** seg\_duan **<=** SEG\_NUM0**;**

4'h1**:** seg\_duan **<=** SEG\_NUM1**;**

4'h2**:** seg\_duan **<=** SEG\_NUM2**;**

4'h3**:** seg\_duan **<=** SEG\_NUM3**;**

4'h4**:** seg\_duan **<=** SEG\_NUM4**;**

4'h5**:** seg\_duan **<=** SEG\_NUM5**;**

4'h6**:** seg\_duan **<=** SEG\_NUM6**;**

4'h7**:** seg\_duan **<=** SEG\_NUM7**;**

4'h8**:** seg\_duan **<=** SEG\_NUM8**;**

4'h9**:** seg\_duan **<=** SEG\_NUM9**;**

4'ha**:** seg\_duan **<=** SEG\_NUMA**;**

4'hb**:** seg\_duan **<=** SEG\_NUMB**;**

4'hc**:** seg\_duan **<=** SEG\_NUMC**;**

4'hd**:** seg\_duan **<=** SEG\_NUMD**;**

4'he**:** seg\_duan **<=** SEG\_NUME**;**

4'hf**:** seg\_duan **<=** SEG\_NUMF**;**

**default:** **;**

endcase

**//????**

always @**(**cnt\_4**[**7**:**6**])**

**case(**cnt\_4**[**7**:**6**])**

2'b00**:** seg\_wei **<=** SEG\_WE0**;**

2'b01**:** seg\_wei **<=** SEG\_WE1**;**

2'b10**:** seg\_wei **<=** SEG\_WE2**;**

2'b11**:** seg\_wei **<=** SEG\_WE3**;**

**default:** seg\_wei **<=** 8'b0000\_0000**;**

endcase

**//**-------------------------------------------------

**//**74HC95**????**

reg ds\_stcpr**;** **//**74HC595**????????????????????????**

reg ds\_shcpr**;** **//**74HC595**?????????????????????**

reg ds\_datar**;** **//**74HC595**???????**

**//????????**

always @**(**posedge clk\_div\_2 or negedge rst\_n**)**

**if(!**rst\_n**)** ds\_shcpr **<=** 1'b0**;**

**else** **if((**cnt\_4 **>** 8'h02 **&&** cnt\_4 **<=** 8'h22**)** **||** **(**cnt\_4 **>** 8'h24 **&&** cnt\_4 **<=** 8'h44**)**

**||** **(**cnt\_4 **>** 8'h46 **&&** cnt\_4 **<=** 8'h66**)** **||** **(**cnt\_4 **>** 8'h68 **&&** cnt\_4 **<=** 8'h88**)**

**||** **(**cnt\_4 **>** 8'h90 **&&** cnt\_4 **<=** 8'hb0**)** **||** **(**cnt\_4 **>** 8'hb2 **&&** cnt\_4 **<=** 8'hd2**)**

**||** **(**cnt\_4 **>** 8'hd4 **&&** cnt\_4 **<=** 8'hf4**))**

ds\_shcpr **<=** **~**ds\_shcpr**;**

**else** ds\_shcpr**<=**1'b0**;**

**//????????**

always @**(**posedge clk\_div\_2 or negedge rst\_n**)**

**if(!**rst\_n**)** ds\_datar **<=** 1'b0**;**

**else**

**case(**cnt\_4**)**

8'h02**,**8'h46**,**8'h90**,**8'hd4**:** ds\_datar **<=** seg\_duan**[**7**];**

8'h04**,**8'h48**,**8'h92**,**8'hd6**:** ds\_datar **<=** seg\_duan**[**6**];**

8'h06**,**8'h4a**,**8'h94**,**8'hd8**:** ds\_datar **<=** seg\_duan**[**5**];**

8'h08**,**8'h4c**,**8'h96**,**8'hda**:** ds\_datar **<=** seg\_duan**[**4**];**

8'h0a**,**8'h4e**,**8'h98**,**8'hdc**:** ds\_datar **<=** seg\_duan**[**3**];**

8'h0c**,**8'h50**,**8'h9a**,**8'hde**:** ds\_datar **<=** seg\_duan**[**2**];**

8'h0e**,**8'h52**,**8'h9c**,**8'he0**:** ds\_datar **<=** seg\_duan**[**1**];**

8'h10**,**8'h54**,**8'h9e**,**8'he2**:** ds\_datar **<=** seg\_duan**[**0**];**

8'h12**,**8'h56**,**8'ha0**,**8'he4**:** ds\_datar **<=** seg\_wei**[**0**];**

8'h14**,**8'h58**,**8'ha2**,**8'he6**:** ds\_datar **<=** seg\_wei**[**1**];**

8'h16**,**8'h5a**,**8'ha4**,**8'he8**:** ds\_datar **<=** seg\_wei**[**2**];**

8'h18**,**8'h5c**,**8'ha6**,**8'hea**:** ds\_datar **<=** seg\_wei**[**3**];**

8'h1a**,**8'h5e**,**8'ha8**,**8'hec**:** ds\_datar **<=** seg\_wei**[**4**];**

8'h1c**,**8'h60**,**8'haa**,**8'hee**:** ds\_datar **<=** seg\_wei**[**5**];**

8'h1e**,**8'h62**,**8'hac**,**8'hf0**:** ds\_datar **<=** seg\_wei**[**6**];**

8'h20**,**8'h64**,**8'hae**,**8'hf2**:** ds\_datar **<=** seg\_wei**[**7**];**

8'h24**,**8'h68**,**8'hb2**,:** ds\_datar **<=** 1**;**

8'h26**,**8'h6a**,**8'hb4**,:** ds\_datar **<=** 1**;**

8'h28**,**8'h6c**,**8'hb6**,:** ds\_datar **<=** 1**;**

8'h2a**,**8'h6e**,**8'hb8**,:** ds\_datar **<=** 1**;**

8'h2c**,**8'h70**,**8'hba**,:** ds\_datar **<=** 1**;**

8'h2e**,**8'h72**,**8'hbc**,:** ds\_datar **<=** 1**;**

8'h30**,**8'h74**,**8'hbe**,:** ds\_datar **<=** 1**;**

8'h32**,**8'h76**,**8'hc0**,:** ds\_datar **<=** 1**;**

8'h34**,**8'h78**,**8'hc2**,:** ds\_datar **<=** 1**;**

8'h36**,**8'h7a**,**8'hc4**,:** ds\_datar **<=** 1**;**

8'h38**,**8'h7c**,**8'hc6**,:** ds\_datar **<=** 1**;**

8'h3a**,**8'h7e**,**8'hc8**,:** ds\_datar **<=** 1**;**

8'h3c**,**8'h80**,**8'hca**,:** ds\_datar **<=** 1**;**

8'h3e**,**8'h82**,**8'hcc**,:** ds\_datar **<=** 1**;**

8'h40**,**8'h84**,**8'hce**,:** ds\_datar **<=** 1**;**

8'h42**,**8'h86**,**8'hd0**,:** ds\_datar **<=** 1**;**

**default:** ds\_datar **<=** seg\_duan**[**0**];**

endcase

**//????????**

always @**(**posedge clk1 or negedge rst\_n**)**

**if(!**rst\_n**)** ds\_stcpr **<=** 1'b0**;**

**else** **if((**cnt\_4 **==** 8'h02**)** **||** **(**cnt\_4 **==** 8'h23**)** **||** **(**cnt\_4 **==** 8'h45**)** **||** **(**cnt\_4 **==** 8'h67**)** **||** **(**cnt\_4 **==** 8'h89**)||** **(**cnt\_4 **==** 8'hb1**)||** **(**cnt\_4 **==** 8'hd3**))** ds\_stcpr **<=** 1'b0**;**

**else** **if((**cnt\_4 **==** 8'h22**)** **||** **(**cnt\_4 **==** 8'h44**)** **||** **(**cnt\_4 **==** 8'h66**)** **||** **(**cnt\_4 **==** 8'h88**)** **||** **(**cnt\_4 **==** 8'hb0**)||** **(**cnt\_4 **==** 8'hd2**)||** **(**cnt\_4 **==** 8'hf4**))** ds\_stcpr **<=** 1'b1**;**

wire ds\_stcp **=** ds\_stcpr**;**

wire ds\_shcp **=** ds\_shcpr**;**

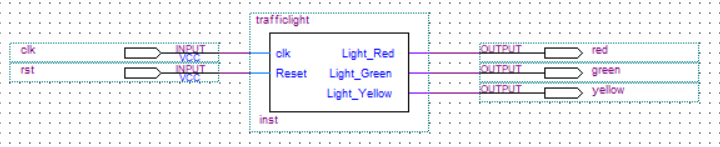
wire ds\_data **=** ds\_datar**;**

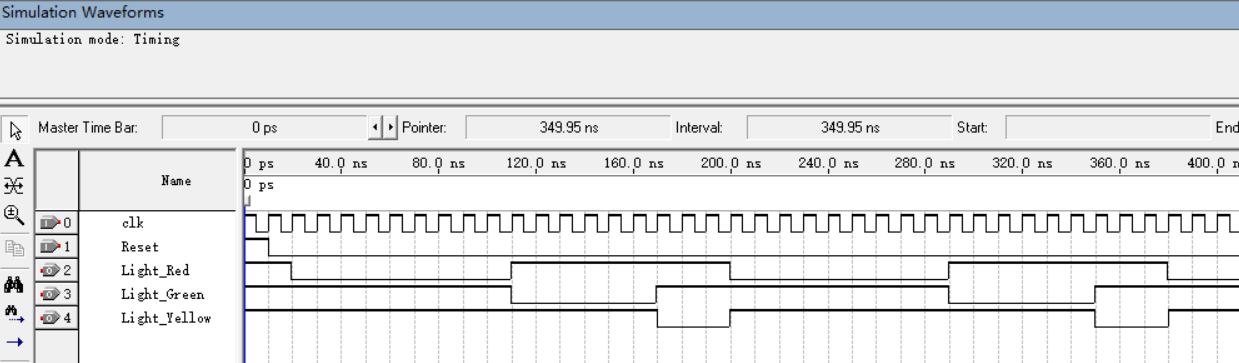
end module

**（实验报告作品相片粘贴页）**

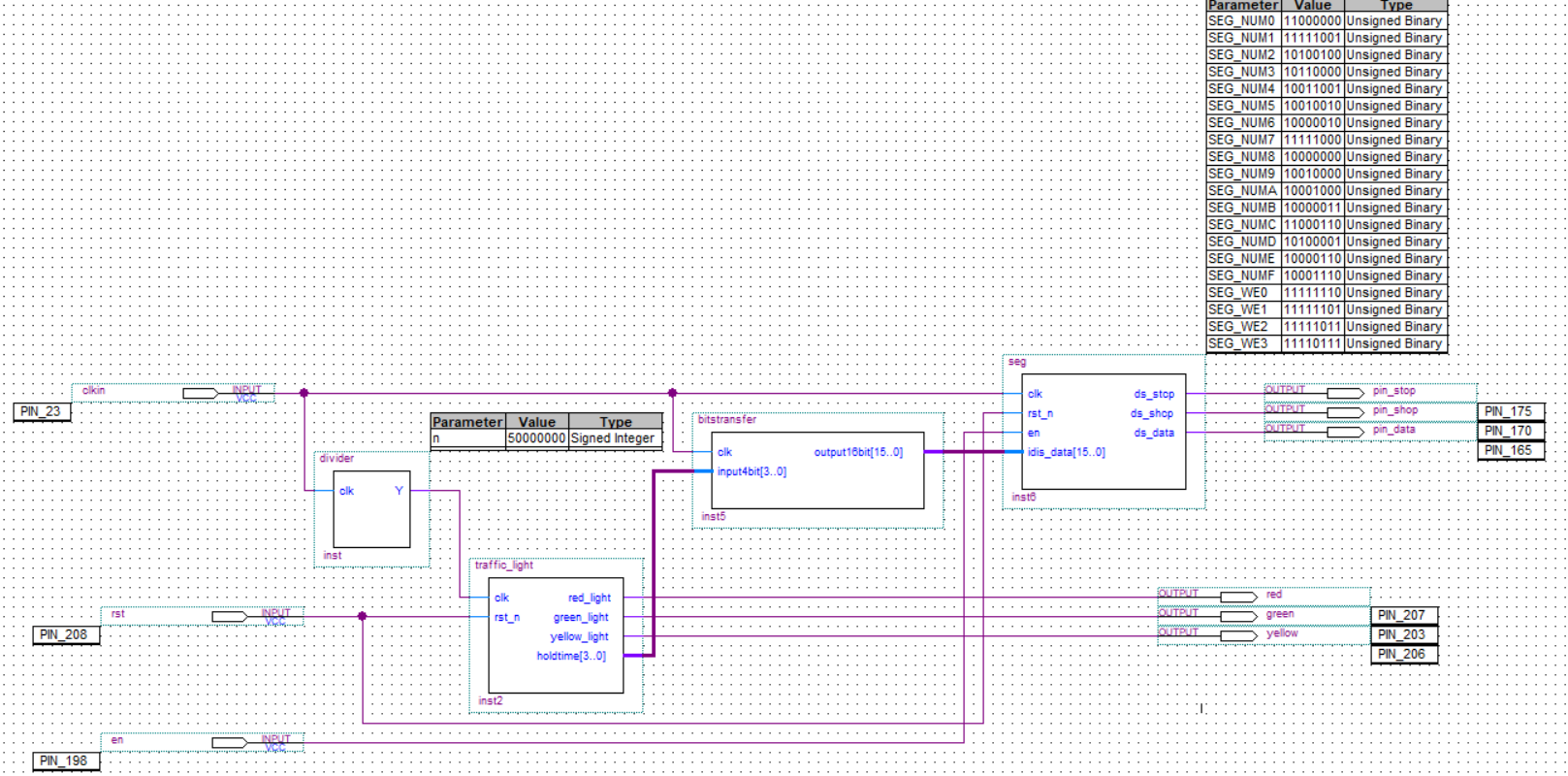
（总电路图截图、仿真结果或者实验平台运行效果照片）

Led低电平时亮灯





Led&数码管

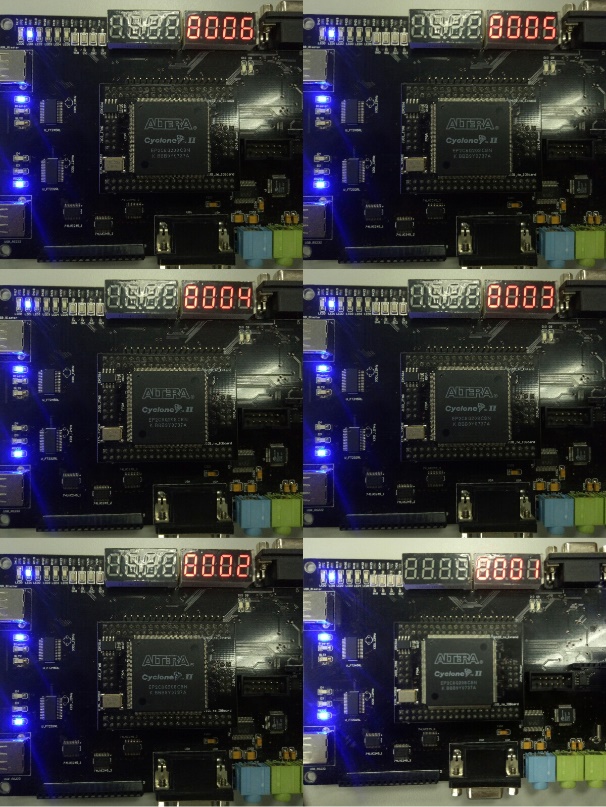
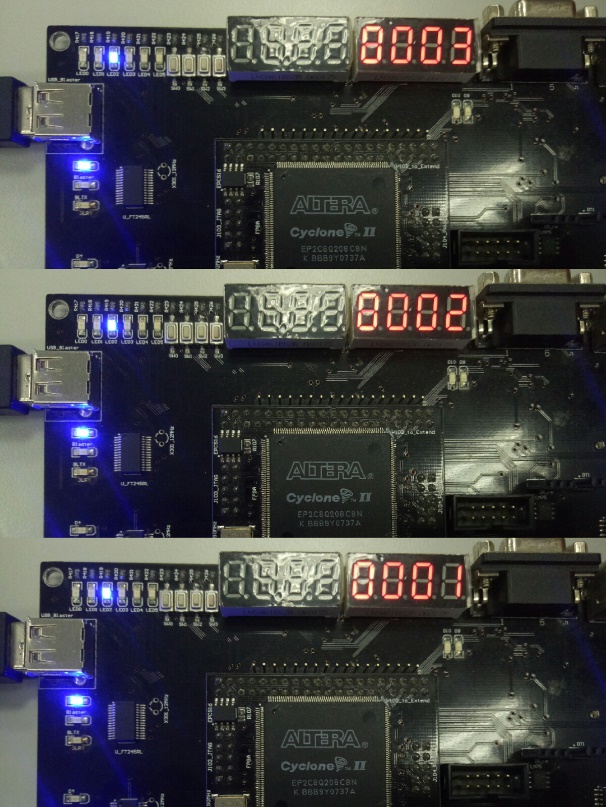


下载后开发板图片：

Red：9s



Green：6s Yellow：3s

# 实验四 按键控制的状态机设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | 31号楼 | 312房； | **实验台号：** | 23 |
| **实验日期与时间：** | 2018.11.28 | | **评 分：** |  |
| **预习检查纪录：** |  | | **批改教师：** |  |

报告内容：

1. 实验目的：

运用状态机设计按键控制数码管显示的电路系统

1. 实验内容：

1、 按键控制数码管显示，当按下S2时，Q4数码管显示1；再次按下S2，Q4数码管显示2；第三次按下S2，Q4数码管显示3；第四次按下S2时，Q4数码管显示4.第五次按下S2时Q4又从1开始显示，如此反复。

2、 检测按键是否按下需要消抖，消抖程序请同学们参考附录程序，参考程序是利用四个按键控制四个LED灯的亮灭。请同学们在读懂参考程序的基础上将其改编为符合要求1的程序。

3、 参考程序是普通的控制程序，本实验要求同学们利用状态机来编写程序。

1. 实验设计原理；

1、八段 LED 译码列表见表 1。

表 1 八段 LED 译码列表（0-4）

|  |  |
| --- | --- |
| 数值 | A B C D E F G H |
| 0 | 0 0 0 0 0 0 1 0 |
| 1 | 1 0 0 1 1 1 1 0 |
| 2 | 0 0 1 0 0 1 0 0 |
| 3 | 0 0 0 0 1 1 0 0 |
| 4 | 1 0 0 1 1 0 0 0 |

2、按键消抖原理

在按键按下或者释放时都会出现一个不稳定的抖动时间，其抖动期一般为5~10ms，见图 1。如果不处理好这个抖动时间，我们就无法采集到正确有效的按键值，所以我们的设计中必须有效地消除按键抖动。如何进行有效的消抖，是本实验的重点。

图 1 按键电平抖动示意图



在按键采集中，为了有效地滤除按键抖动，使用了个20 ms的计数器，在按键值没有变化时，这个计数器总是不停地计数，并且计数到20 ms最大值时进行一次当前按键值采样(作为最终键值锁存下来)。

另外，还专门设置两个寄存器(实际上为获得更稳定的采样结果，使用了 4 个寄存器进行当前键值的采样缓存，而用于内部判断的为最后 2 级的寄存器)对当前的按键输入值进行多拍锁存(并不作为最终的键值)，并且利用这两个寄存器前值的变化来判断当前键值是否有跳变（如从 1 变成 0 或从 0 变成 1)。若有键值的跳变.则 20 ms计数器就会清 0，相当于重新开始计数，这样就能够保证按键被按下或者松开时短于 20 ms的抖动情况下不锁存键值，从而达到滤除任何短于20 ms 的按键抖动。

本次实验的关键在与按键消抖部分程序的编写，为了有效地滤除按键抖动，采用脉冲边缘检测法来检测按键的抖动，代码的具体实现为：

key\_rst\_an **<=** key\_rst\_r and not key\_rst**;**

low\_sw\_an **<=** low\_sw\_r and not low\_sw**;**

脉冲边缘检测法检测按键抖动的原理如下

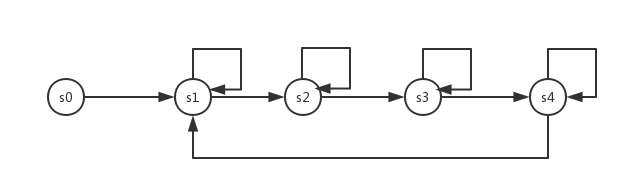
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| key\_rst | 1 | 1 | 1 | 0 | 0 | 1 |  |
| Not key\_rst | 0 | 0 | 0 | 1 | 1 | 0 |  |
| key\_rst\_r |  | 1 | 1 | 1 | 0 | 0 | 1 |
| key\_an |  | 0 | 0 | 1 | 0 | 0 |  |

当 key\_rst 电平从 1 变为 0 时，key\_an 将变为 1。以 key\_an 的值来判断检测按键是否抖动。

编写出按键控制数码管显示程序后，应对该模块进行仿真，观察输入与输出的关系是否正确，但由于 Quartus II 软件自带的仿真软件仿真时间限制，本次实验直接进行整个工程的结合，并且编译和下载至开发板上

四, 实验过程记录（流程图或者实验逻辑思路过程）

状态转换图：



五，实验结果和分析

将工程下载至开发板上，每隔 2 秒按 1 次 SW2 按键，可以得到数码管显示0,1,2,3,4,1,2,3,4……符合实验要求。

六，附录（打印源代码页）

--keycontrol code

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**use** ieee**.**std\_logic\_arith**.all;** --including conv\_std\_logic\_vector

**entity** keycontrol **is**

**port(**clk**:in** std\_logic**;**

sw**:in** std\_logic**;**

seven\_seg\_led**:out** std\_logic\_vector**(**15 **downto** 0**)**

**);**

**end** **entity;**

**architecture** behav **of** keycontrol **is**

**signal** key\_rst**:**std\_logic**;**

**signal** key\_rst\_r**:**std\_logic**;**

**signal** key\_rst\_an**:**std\_logic**;**

**signal** low\_sw**:**std\_logic**;**

**signal** low\_sw\_r**:**std\_logic**;**

**signal** low\_sw\_an**:**std\_logic**;**

**signal** mtime**:**integer **range** 0 **to** 9**;**

**signal** cnt**:**std\_logic\_vector**(**19 **downto** 0**);**

-- five state

**type** state\_type **is** **(**s0**,** s1**,** s2**,** s3**,** s4**);**

**signal** pre\_state**,** nx\_state**:**state\_type**;** --xiantai he citai

**begin**

--chuancan1\_1

**process(**clk**)**

**begin**

**if(rising\_edge(**clk**))then**

key\_rst**<=**sw**;**

**end** **if;**

**end** **process;**

--chuancan1\_2

**process(**clk**)**

**begin**

**if(rising\_edge(**clk**))then**

key\_rst\_r**<=**key\_rst**;**

**end** **if;**

**end** **process;**

key\_rst\_an **<=** key\_rst\_r and not key\_rst**;** --xiaodou

--jishu

**process(**clk**)**

**begin**

**if(rising\_edge(**clk**))then**

**if(**key\_rst\_an**=**'0'**)then**

cnt**<=**cnt**+**1**;**

**else** cnt**<=(others=>**'0'**);**

**end** **if;**

**end** **if;**

**end** **process;**

--chuancan2\_1

**process(**clk**)**

**begin**

**if(rising\_edge(**clk**))then**

**if** cnt**=**"11111111111111111111" **then**

low\_sw**<=**sw**;**

**else** **null;**

**end** **if;**

**end** **if;**

**end** **process;**

--chuancan2\_2

**process(**clk**)**

**begin**

**if(rising\_edge(**clk**))then**

low\_sw\_r**<=**low\_sw**;**

**end** **if;**

**end** **process;**

low\_sw\_an **<=** low\_sw\_r and not low\_sw**;** --xiaodou

-- seq process:create state

seq**:process(**clk**)**

**begin**

**if(rising\_edge(**clk**))then**

pre\_state **<=** nx\_state**;**

**end** **if;**

**end** **process;**

-- com process:create next state and output logic

com**:process(**pre\_state**,**low\_sw\_an**)**

**begin**

**case** pre\_state **is**

**when** s0**=>**

mtime**<=**0**;**

seven\_seg\_led **<=** conv\_std\_logic\_vector**(**mtime**,**16**);** --convert mtime as a vector with the length of 16

**if** low\_sw\_an**=**'1' **then** nx\_state**<=**s1**;**

**else** nx\_state**<=**s0**;**

**end** **if;**

**when** s1**=>**

mtime**<=**1**;**

seven\_seg\_led **<=** conv\_std\_logic\_vector**(**mtime**,**16**);** --convert mtime as a vector with the length of 16

**if** low\_sw\_an**=**'1' **then** nx\_state**<=**s2**;**

**else** nx\_state**<=**s1**;**

**end** **if;**

**when** s2**=>**

mtime**<=**2**;**

seven\_seg\_led **<=** conv\_std\_logic\_vector**(**mtime**,**16**);** --convert mtime as a vector with the length of 16

**if** low\_sw\_an**=**'1' **then** nx\_state**<=**s3**;**

**else** nx\_state**<=**s2**;**

**end** **if;**

**when** s3**=>**

mtime**<=**3**;**

seven\_seg\_led **<=** conv\_std\_logic\_vector**(**mtime**,**16**);** --convert mtime as a vector with the length of 16

**if** low\_sw\_an**=**'1' **then** nx\_state**<=**s4**;**

**else** nx\_state**<=**s3**;**

**end** **if;**

**when** s4**=>**

mtime**<=**4**;**

seven\_seg\_led **<=** conv\_std\_logic\_vector**(**mtime**,**16**);** --convert mtime as a vector with the length of 16

**if** low\_sw\_an**=**'1' **then** nx\_state**<=**s1**;**

**else** nx\_state**<=**s4**;**

**end** **if;**

**end** **case;**

**end** **process** com**;**

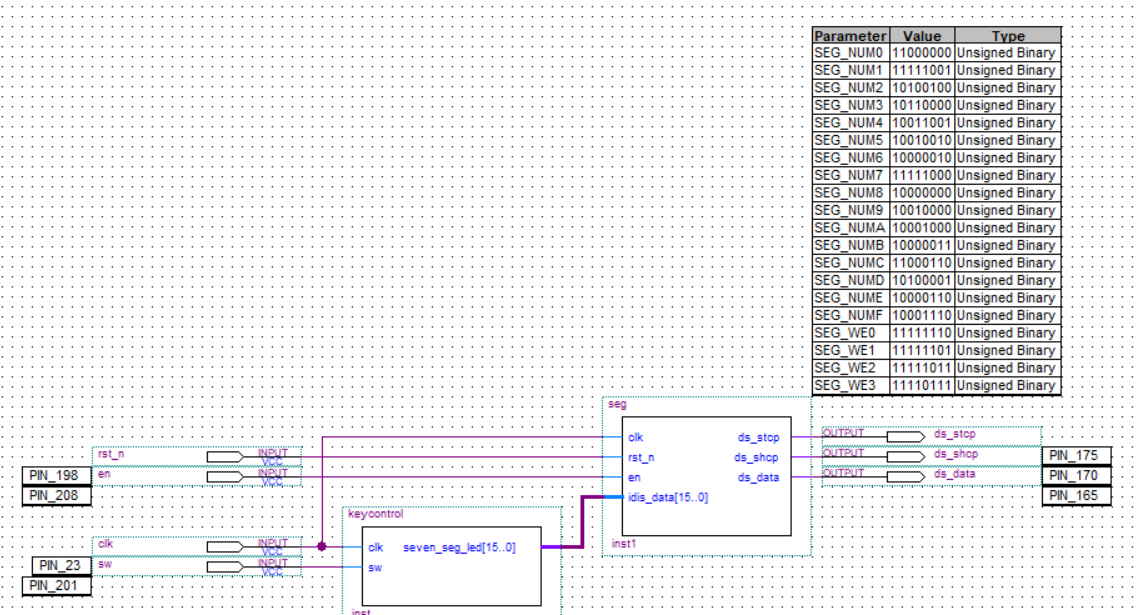
**end** behav**;**

//7 segment tube 调用上一实验中的数码管驱动模块

**（实验报告作品相片粘贴页）**

（总电路图截图、仿真结果或者实验平台运行效果照片）

总电路图：



开发板照片0,1,2,3,4,1,2,3,4



**学生工作日志（由参与实验同学填写）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | | **时间** | | **地点** | **主要工作内容** |
| 1 | | 2018年10月17日 | | 310312 | QUARTUS软件的使用；VHDL程序结构的熟悉 |
| 2 | | 2018年10月31日 | | 310312 | 基本电路模块设计（组合、时序） |
| 3 | | 2018年11月14日 | | 310312 | 基于状态机的交通灯控制 |
| 4 | | 2018年11月28日 | | 310312 | 按键控制的状态机设计 |
| 5 | …… | |  | |  |