

数字电子技术综合实验报告

**题 目 基于**FPGA**的多功能创意番茄钟**

**电气 学院 电气工程及其自动化 系**

**学生姓名 聂永欣 班级 电气810 学号 2186113564**

**指导教师 金印斌**

**2020年11月**

**实验题目：基于**FPGA**的多功能创意番茄钟**

**学生姓名：聂永欣**

**指导教师：金印斌**

**摘 要**

本文详细介绍了一款基于FPGA EGO1开发板的创意番茄钟，通过实验完成一款具有25分钟和5分钟倒计时以及可以进行按键控制功能的番茄钟。

实验内容主要包括三大部分：一是番茄钟的基本倒计时功能并可以通过按键暂停并重置，在倒计时结束后，通过数码管的闪烁表示倒计时结束，；二是记录番茄钟的循环周期并使用数码管显示已经进行的周期数；三是能够自动调节倒计时的时间，包括休息时间和工作时间的分钟和秒数的长度，通过按键实现。这一部分主要通过有限状态机实现。

最后，通过Xilinx的7系列FPGA增加了XADC硬核模块与的电位器，加入了调节数码管亮度这一功能，即通过转动电位器控制数码管亮度。

**关 键 词**：基本倒计时；有限状态机；循环周期；XADC硬核模块；亮度控制

**目 录**

[1 前 言 1](#_Toc57972611)

[1.1 设计背景 1](#_Toc57972612)

[1.2 主要实现的功能 1](#_Toc57972613)

[1.3 设计思路 2](#_Toc57972614)

[1.4 设计难点与亮点 2](#_Toc57972615)

[1.4.1 设计难点 2](#_Toc57972616)

[1.4.2 设计亮点 2](#_Toc57972617)

[2 系统设计 4](#_Toc57972618)

[2.1 总体结构及功能设计 4](#_Toc57972619)

[2.1.1 实验平台 4](#_Toc57972620)

[2.1.2 实验设计流程 4](#_Toc57972621)

[2.1.3 输入输出设计 5](#_Toc57972622)

[2.2 子模块设计 5](#_Toc57972623)

[2.2.1 子模块划分 5](#_Toc57972624)

[2.2.2 模数转换模块简介 **错误!未定义书签。**](#_Toc57972625)

[2.2.3 RTL顶层逻辑图 6](#_Toc57972626)

[2.2.4 子模块设计思路 6](#_Toc57972627)

[2.3 硬件占用 7](#_Toc57972628)

[3 系统调试 9](#_Toc57972629)

[3.1 遇到的问题 9](#_Toc57972630)

[3.1.1 倒计时功能发现问题 9](#_Toc57972631)

[3.1.2 其他问题 9](#_Toc57972632)

[3.2 解决方案 10](#_Toc57972633)

[3.2.1 倒计时功能问题解决 10](#_Toc57972634)

[3.2.2 其他问题解决 10](#_Toc57972635)

[3.3 设计改进 11](#_Toc57972636)

[3.3.1 倒计时功能改进方案 11](#_Toc57972637)

[4 系统使用说明 12](#_Toc57972638)

[4.1 数码管 12](#_Toc57972639)

[4.2 按键控制 12](#_Toc57972640)

[4.2.1 开关模块 12](#_Toc57972641)

[4.2.2 按键控制 12](#_Toc57972642)

[4.3 电位器旋钮 13](#_Toc57972643)

[5 参考文献 14](#_Toc57972644)

[6 总结 15](#_Toc57972645)

[6.1 课程收获 15](#_Toc57972646)

[6.2 心得体会 15](#_Toc57972647)

[6.3 实验总结 15](#_Toc57972648)

[7 附 录 17](#_Toc57972649)

[7.1 源程序 17](#_Toc57972650)

[7.1.1 顶层模块top 17](#_Toc57972651)

[7.1.2 模数转换顶层模块XADC\_channel 19](#_Toc57972652)

[7.1.3 模拟信号读取模块read 20](#_Toc57972653)

[7.1.4 番茄钟顶层模块tomato\_clock 20](#_Toc57972654)

[7.1.5 秒脉冲产生模块Sec\_pule\_generator 22](#_Toc57972655)

[7.1.6 开关消抖模块ButtonEdgeDetect 22](#_Toc57972656)

[7.1.7 25分钟倒计时顶层模块counter25\_top 23](#_Toc57972657)

[7.1.8 状态记录模块State 24](#_Toc57972658)

[7.1.9 25分钟倒计时模块counter25 25](#_Toc57972659)

[7.1.10 周期计数信号edgecounter 29](#_Toc57972660)

[7.1.11 微分器differenciator 31](#_Toc57972661)

[7.1.12 按键控制周期pressadjuster 31](#_Toc57972662)

[7.1.13 PWM控制器pwmctrl 32](#_Toc57972663)

[7.2 约束文件 32](#_Toc57972664)

[7.2.1 CON\_of\_tomato 32](#_Toc57972665)

# 前 言

## 设计背景

在信息大爆炸的时代，各路信息狂轰乱炸，人们的注意力被大大分散，各种提高工作效率的工具越来越多，但是很多大学生的工作效率并没有得到提高，反而越来越陷入无法按时完成自己的课业，无法平衡学业和生活的困境，时间一长就越发焦虑和不安，每天一想到上学心情就会很沉重。而番茄工作法是一种简单易行的时间管理方法，可以用来提升学习效率。

番茄工作法是由弗朗西斯科·西里洛于1992年创立的一种相对于GTD（Getting Things Done）更微观的时间管理方法。使用番茄工作法，选择一个待完成的任务，将番茄时间设为25分钟，专注工作，中途不允许做任何与该任务无关的事，直到番茄时钟响起，然后在纸上画一个对号短暂休息一下(5分钟就行)，每完成4个番茄时段就多休息一会儿。

番茄工作法是一套简单的工具和流程，其优点如下：

1．减轻时间焦虑；

2．提升集中力和注意力，减少中断；

3．增强决策意识；

4．唤醒激励和持久激励；

5．巩固达成目标的决心；

6．完善预估流程，精确地保质保量；

7．改进工作学习流程；

8．强化决断力，快刀斩乱麻；

标准番茄钟的定时时间是工作25分钟，休息5分钟，但由于个体的差异，25分钟 /5分钟的标准番茄钟不一定适用任何人。西里洛最早使用的番茄钟是机械式定时器，后来有人用软件设计了这样的定时器安装在电脑或手机中。环境的差异使机械式定时器及电脑定时软件的使用受到约束，如在某些环境下没有电脑、在自习室不应让手机发出声音等等，本实验设计的创意多功能番茄钟作为时间管理的工具却可随心所欲，而且可以在提高使用效率的基础上增加多种人性化功能。

## 主要实现的功能

本实验要求完成一款基于FPGA的多功能创意番茄钟，用于帮助实现番茄工作法，具有以下功能：

1. 启动后为用户显示标准番茄时间和休息时间，即标准番茄时间25分钟，和休息时间5分钟。
2. 在番茄时间和休息时间均为用户提供倒计时的计时方式。
3. 倒计时至全零状态时，增加提示功能（数码管闪烁15秒），然后自动显示下一初值状态。
4. 为用户提供在任意时间终止计时的功能。
5. 为用户提供番茄时间数量的统计功能，即终止计时后用户可以获知已专注于当前工作的番茄数。
6. 为用户提供修改标准番茄时间长度的功能，包括分钟倒计时长和秒计时时长的功能。
7. 可以通过旋钮调节数码管亮度，以减少数码管灯光在工作状态下对使用者的影响。

## 设计思路

有限状态机是是系统实现的中心思想，要想有条不紊地实现程序设计，必须要牢牢。本程序主要由三大模块组成：有限状态机基础倒计时模块、模数转换模块、周期设置模块。这三个模块体现了程序的三部分主要功能。

对于基本的倒计时功能，可以将25分钟计数器和5分钟计时组合，通过状态机的方式转换为可以通过按键控制的计时器。同时，在完成番茄钟基础功能的同时，在计数器到达0状态时，增加一个状态，通过闪烁数码管的形式提醒计时结束。

对于进阶功能，可以通过增加一个变量flag的方式来实现周期的计数，在每个番茄周期结束后。同理，对于可调时间的模式，增加一个状态转化的按键用于在正常番茄钟模式和置数模式中间切换，然后通过另外两个按键分别控制分钟和秒数的加减。

对于创意功能，我认为可以使用实验板上的AD转换模块来完成，通过电位器调节电压产生的模拟量信号转化为数字量信号，将读取的数字量信号输入PWM控制模块，用以调节数码管的亮度。

## 设计难点与亮点

### 设计难点

1. 为用户提供番茄时间周期的统计功能；
2. 不同计数状态之间的转换；
3. 番茄钟的校时功能；
4. 通过ADC和电位器控制数码管的亮度。

### 设计亮点

1. 采用多模块化设计；
2. 设计逻辑清晰，便于修改和增加功能；
3. 利用了实验开发板上的AD模块用以调整数码管亮度。

# 系统设计

## 总体结构及功能设计

### 实验平台

FPGA取自Field Programmable Gate Array这四个英文单词，直译为“现场可编程逻辑阵列”，就是可反复变成的逻辑器件（可编程数字电路）。是在PAL、GAL等可编程器件的基础上进一步发展的产物。它是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。

FPGA 器件属于专用集成电路中的一种半定制电路，是可编程的逻辑列阵，能够有效的解决原有的器件门电路数较少的问题。FPGA的基本结构包括可编程输入输出单元，可配置逻辑块，数字时钟管理模块，嵌入式块RAM，布线资源，内嵌专用硬核，底层内嵌功能单元。

FPGA基本的组成有三大部分：

1. 可配置逻辑块（Configurable Logic Blocks，CLB）；
2. 输入输出模块（Input Output Blocks，IOB）；
3. 内部互联资源（Interconnector）。

而本实验采用的是由依元素科技基于Xilinx Artix-7 FPGA研发的便携式数模混合基础教学实验平台。EGO1配备的FPGA具有大容量高性能的特点，能实现较复杂的数字逻辑。该平台拥有丰富的外设以及灵活的通用拓展接口。

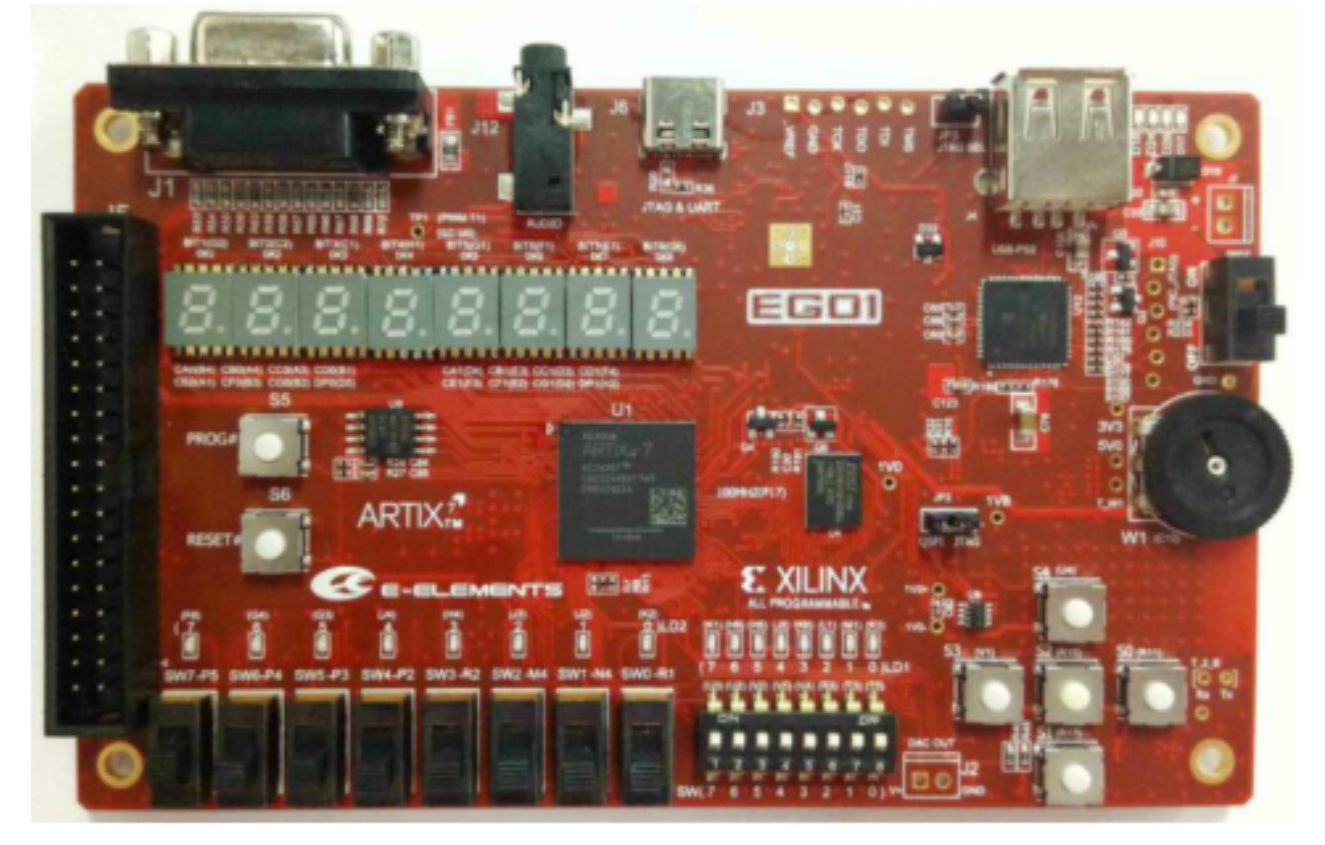


图2.1 EGO1开发板

### 实验设计流程

首先，设计输入阶段创建FPGA工程，添加源文件等；然后是设计综合阶段，FPGA开发工具的综合引擎将编译整个设计，并将HDL源文件转译为特定结构的设计网表；接下来是约束输入，通过指定番茄钟时序以及布线布局；这些过程中穿插设计仿真，对工程进行功能或时序验证。再之后是设计实现，将逻辑设计进一步转译为目标器件中的特定物理文件格式，即生成bitstream文件；然后对结果进行分析，并对资源占用率，结果功耗等进行分析；接着提出设计优化，进行修改，不断优化结果，重新设计综合，约束输入，再一次实现设计，最后板级调试。

### 输入输出设计

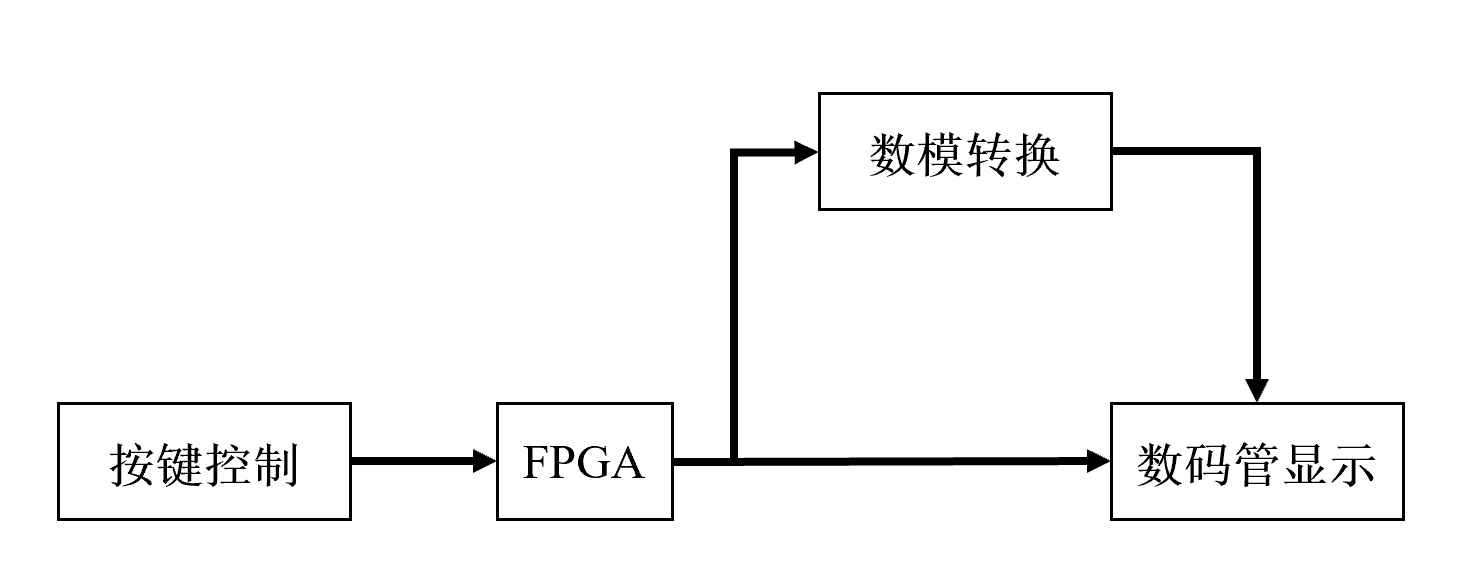


图2.2 系统原理框图

本实验设计了3个输入模块，分别为数模转换模块，番茄钟基础功能模块，自定义周期模块，共有九个输入变量，分别为时钟信号，控制番茄时间暂停，番茄时间复位，番茄钟功能转换，番茄钟自定义周期增加分钟，番茄钟自定义周期增加秒，模拟电压信号（控制数码管亮度）。共有两类输出，分别为数码管亮灭的控制信号和数码管的亮度控制信号。

## 子模块设计

### 子模块划分

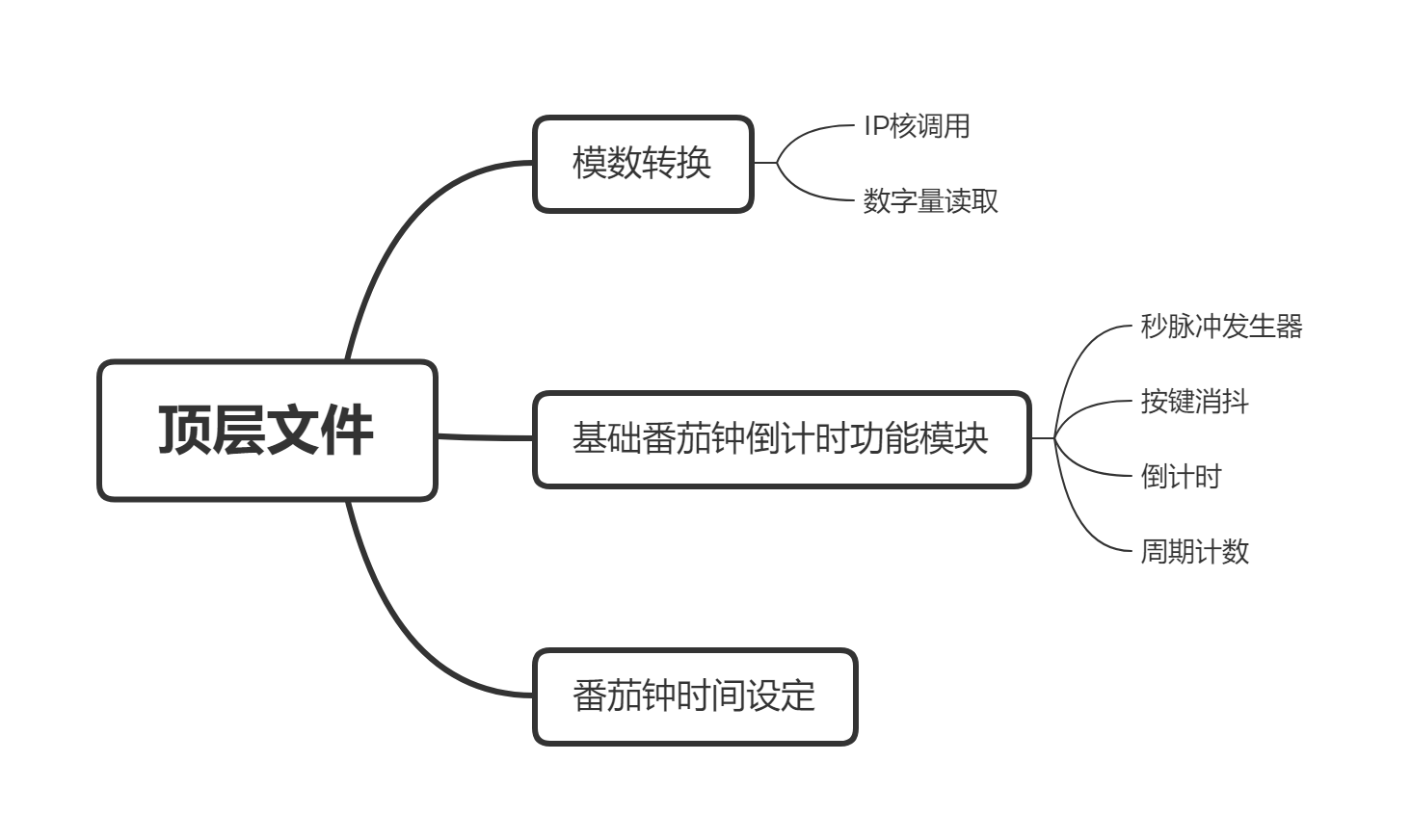


图2.3子模块划分示意图

### RTL顶层逻辑图

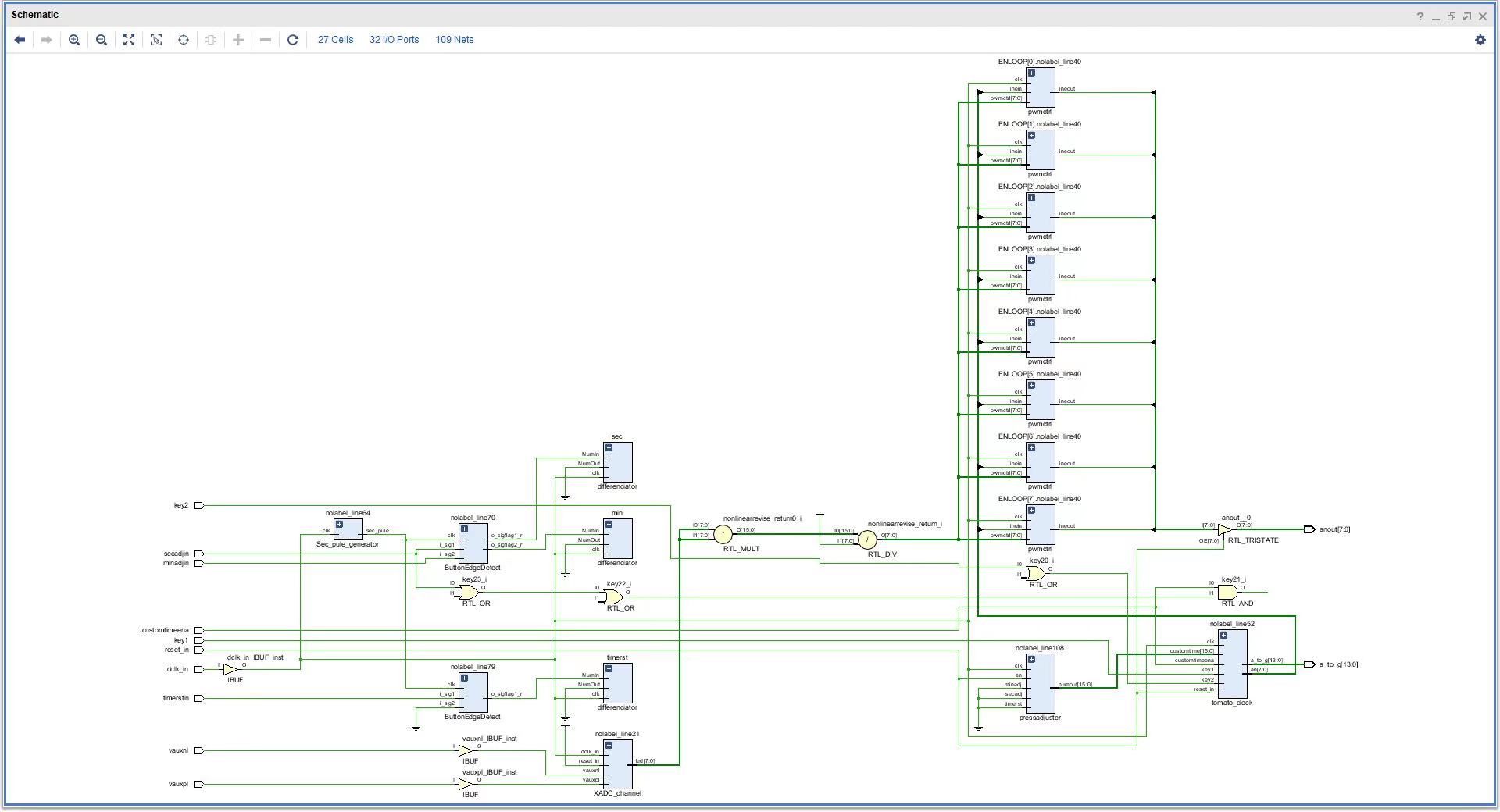


图2.6 RTL分析顶层逻辑图

### 子模块设计思路

##### 数模转换模块设计思路

Xilinx的7系列FPGA增加了XADC硬核模块，为各种不同的应用提供了通用、高精度的模拟接口。提高系统集成度。XADC包括了双12位，1MSPS（Mega sample per second）采样率的ADC，以及片上传感器。在默认模式下，可以采集芯片温度和内部各个供电电压，用于监控FPGA内部状况。同时，ADC具有17个外部模拟输入通道（17对引脚），一对专用模拟差分输入对VP/VN，16对可复用的辅助模拟输入引脚ADXP/ADXN，不使用时可作为通用I/O使用。

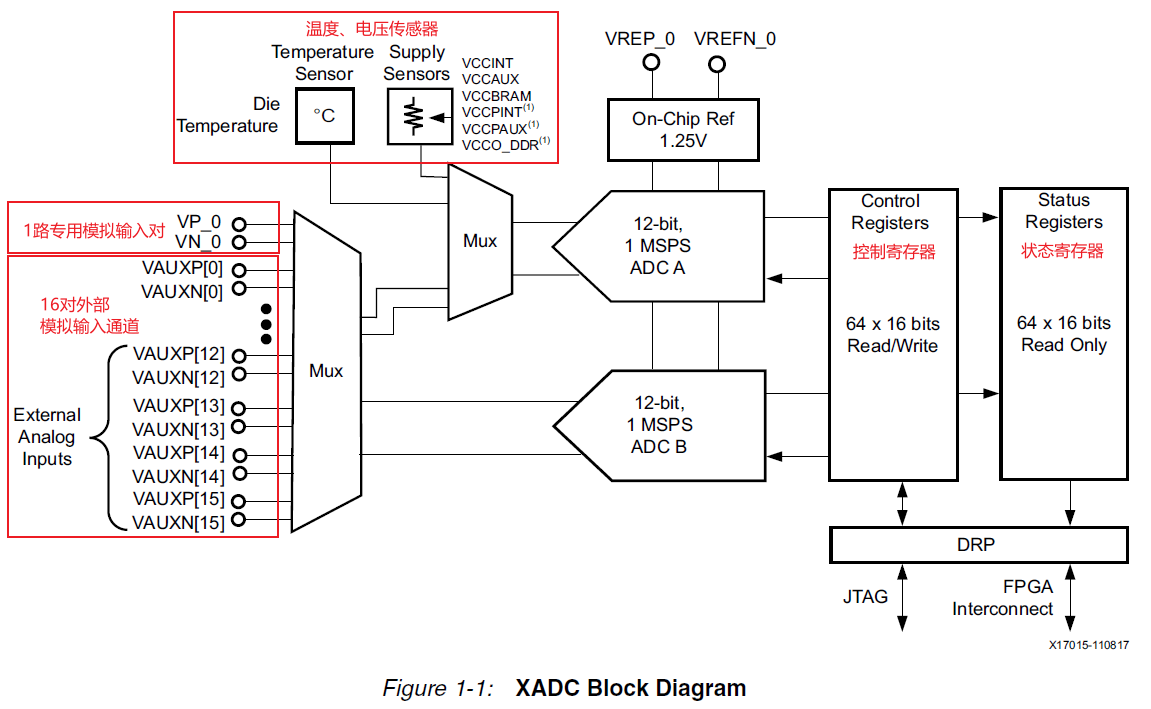


图2.4 XDAC原理

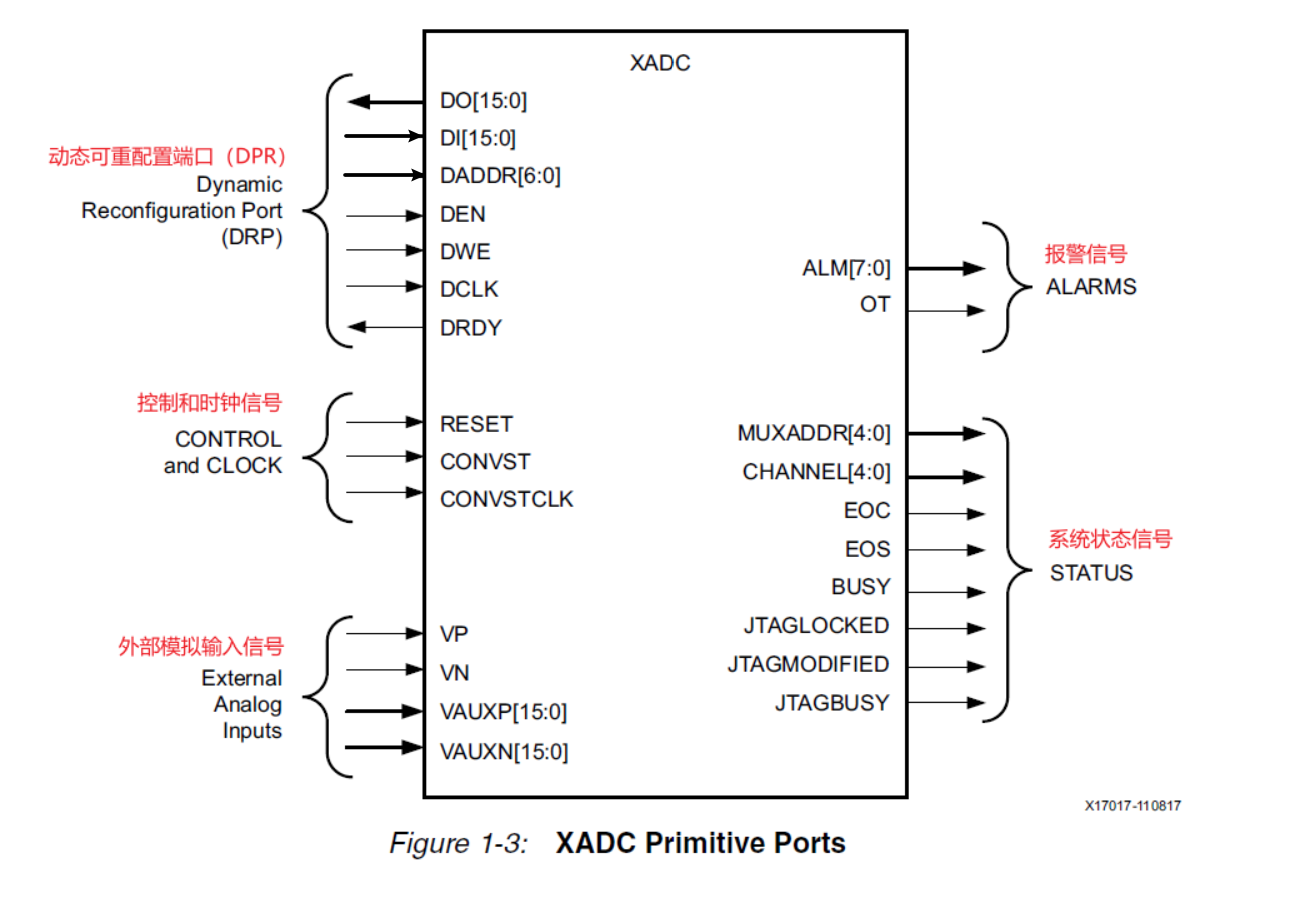


图2.5 XDAC模块框图

首先，通过IP目录添加XADC IP核到设计中，通过GUI对XADC Wizard IP核模块进行配置。根据用户设置，自动配置相关寄存器。然后，新建一个读取模块用于读取模拟信号转化的数字信号，在本项目中为read模块，然后将读取的信号导入PWM模块中用以调节数码管的亮度。

##### 番茄钟基本模块设计思路

番茄钟的基础功能通过有限状态机的方法实现，基础功能为一个米勒型（Mealy）状态机。一个always使用同步时序描述状态转移，一个always使用组合逻辑判断状态转移的条件，描述状态转移的规律，最后在一个或多个always中采用同步时序描述状态的输出。

##### 自定义周期模块设计思路

通过一个开关键转化为自定义周期模式，然后加入一个新的状态，用于自定义周期，此状态与基础功能状态互不干扰。

## 硬件占用

由下图可见本实验的硬件占用情况：

由Imlementation report可见硬件占用情况较好，查找表LUT和触发器FF分别仅占用2%和1%，IO占用情况为15%。

综上所述，该实验的占用情况较好，说明代码逻辑较为清晰，所占用的硬件较少，但仍有优化的空间。

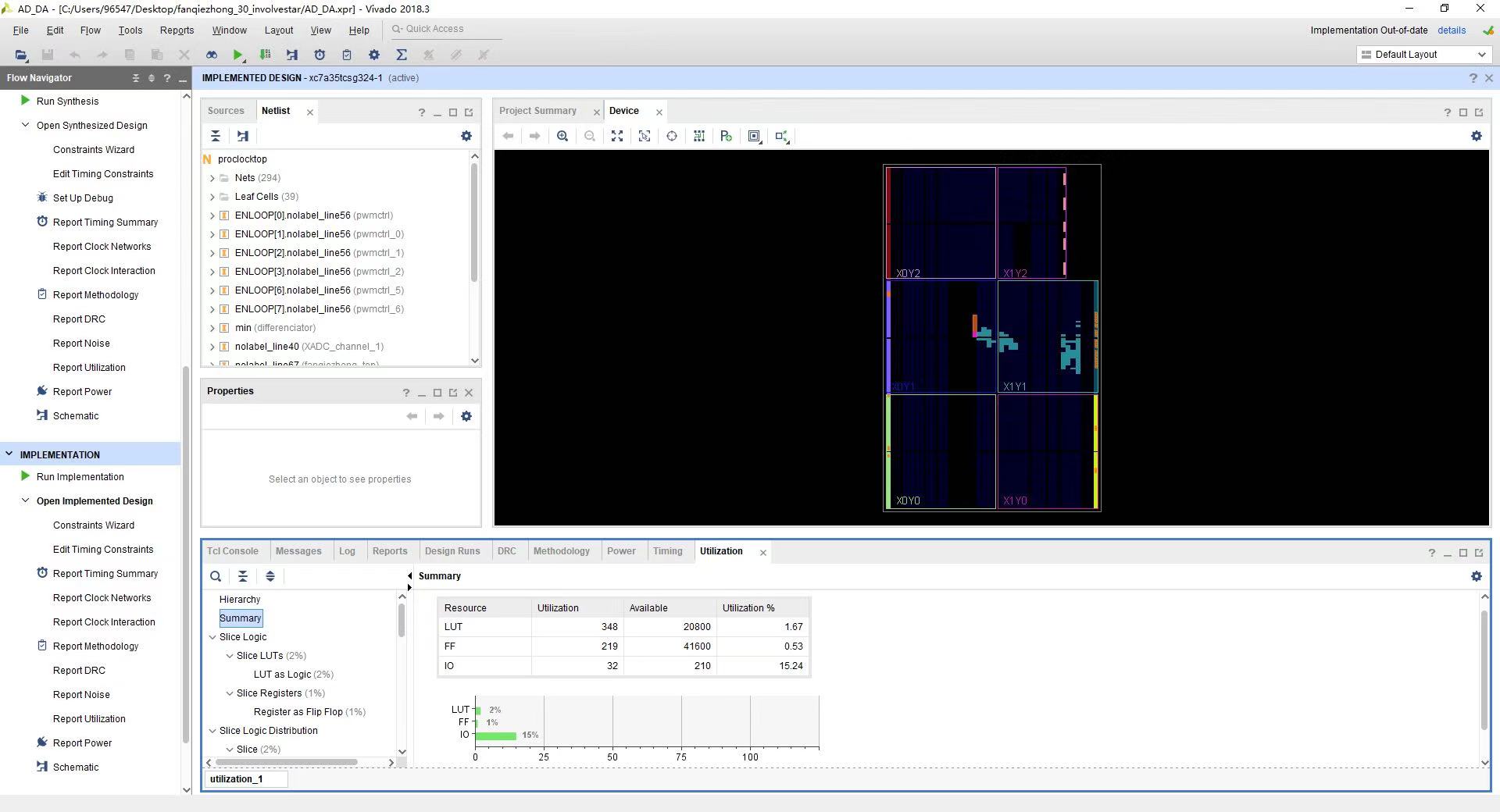


图2.7 Imlementation report

# 系统调试

## 遇到的问题

### 倒计时功能发现问题

##### 初值显示不正常

在25分钟倒计时状态中，初值显示不正常。

##### 秒显示不正常

在25分钟倒计时状态中，秒显示不正常，秒变化过程中会引起分钟的变化，分钟状态不能稳定。

### 其他问题

##### 查看RTL级原理图时报错

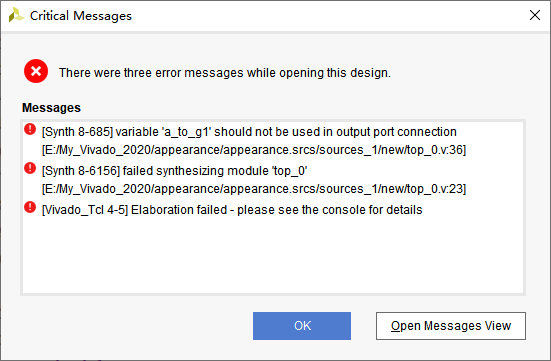


图3.1 报错信息

##### 板子无法连接

hardware窗口显示localhost(0)

##### 整合时报错

ERROR:HDLCompilers:246 - "\*.v" line \* Reference to vector reg '\*' is not a legal net lvalue

ERROR:HDLCompilers:53 - "\*.v" line \* Illegal left hand side of continuous assign

## 解决方案

### 倒计时功能问题解决

##### 初值显示不正常

解决方案：倒计时计数器里的rst信号不统一，统一改为negedge触发，低有效。复位中使用了阻塞赋值，不能在一个always中即用“=”，又用“<=”。这是导致初始化不正常的主要原因。

##### 秒显示不正常

解决方案：borrow信号，在else中没有赋值。此时，当pulse\_in=1时x=0，borrow=1，在下一个pulse\_in来之前，borrow会一直保持高，也就是，borrow脉宽为1s，而不是1个clk，因此，分位会连续变化。

### 其他问题解决

##### 查看RTL原理时报错

解决方案

1. 删除顶层中硬件描述语句，只能用于连接模块；
2. 删除顶层中的reg型信号，改为是wire型信号。

##### 板子无法连接

确保开关打开，在localhost（0）右键选则“close server”，然后从新连接。

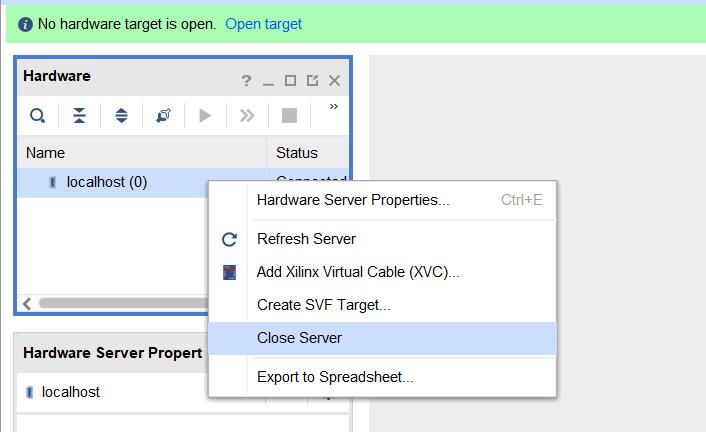


图3.2 解决方法

##### 整合时报错

assign reg型变量，reg型变量不能使用assign命令。

##### 数码管引脚问题 signal is connected to multiple drivers

输出变量在四个语句块中都进行了赋值，应该分别对4个变量进行复制，最后再通过条件选择其中一个

## 设计改进

### 倒计时功能改进方案

1. 检查所有模块clk和rst是否正确，统一。即，rst是否都为统一的上升沿触发（高有效），下降沿触发（低有效）。
2. 检查所有模块，always描述时序逻辑时，所有赋值均使用“<=”；组合逻辑中，所有赋值均为“=”。
3. 为了避免上述问题，每个个always块描述一个信号。甚至可以将x[7:4]和x[3:0]分别在两个always中描述。

# 系统使用说明

## 数码管

数码管用于显示周期数和倒计时的所剩余的时间，同时具有提示休息时间结束的功能。

数码管x1和x2用于显示目前已经运行的周期数，而数码管x2和x3用于显示番茄时间的剩余分钟数，数码管x2和x3用于显示番茄时间的剩余秒数。

在番茄时间和休息时间结束后，所有数码管（x1，x2，x3，x4，x5，x6）会闪烁一段时间用于提示休息时间的结束。

## 按键控制

### 开关模块

开关s1是番茄钟的开关，在打开实验板FPGA的总开关并写入bitstream文件之后，需要打开开关s2，这时，可以看到数码管亮起，这种现象代表实验板为番茄钟运行模式。

开关b3为状态转换按键。当打开开关b3时，番茄钟由基础番茄钟倒计时功能模式转化为自定义周期倒计时模式，再次打开开关b3时，番茄钟由自定义周期倒计时模式转化为基础番茄钟倒计时模式。打开开关b3，番茄钟处于自定义周期状态。

### 按键控制

数码管初始状态为25分钟番茄时间与5分钟休息时间的基本功能状态（25分钟番茄钟时间+5分钟休息时间）。

按键b1为暂停按键。当按下按键b1时，番茄钟开始倒计时，再次按下按键b1时，番茄钟倒计时暂停，此时番茄钟处于暂停状态。不论处于番茄钟基本功能状态还是自定义周期状态。

按键b2为复位按键。当按下按键b2时，番茄钟回到初始状态（当番茄钟为基础倒计时时回到25分钟倒计时状态，当番茄钟为自设定倒计时状态时，回到用户设置的番茄钟倒计时间）。按键b2只有在暂停状态下才能被复位。

按键b4和按键b5为自设定周期倒计时中的分钟调整模块和秒调整模块，当番茄钟处于番茄钟自定义周期模式下的暂停状态下时，按下按键b4，自定义周期分钟数加一，按下按键b5自定义周期秒数加一。分钟数上限为99，秒数上限为59。当分钟数被设定为99时再按下按键b4，时间由99变为0，同理，当秒数被设定为59时，再次按下按键b5，时间由59变为0。

## 电位器旋钮

当开关s1打开时，数码管亮起，可以通过旋转电位器x3实现数码管亮度的控制，在任何状态下都可以改变数码管的亮度，顺时针旋转为亮度增大，逆时针旋转为亮度减小。

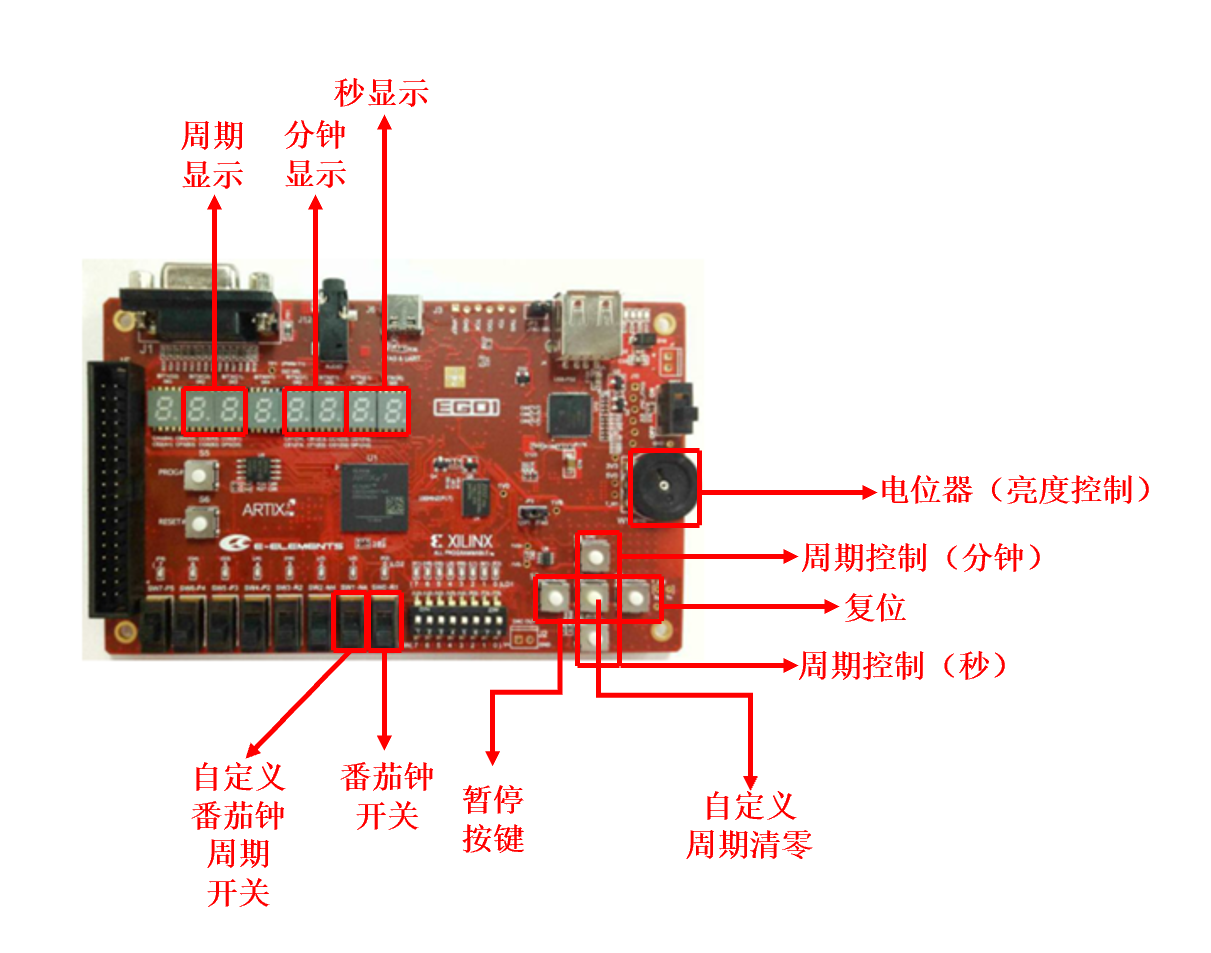


图4.1 番茄钟功能按键示意图

# 参考文献

[1]杨柳,李百伦,徐杨,李鹏飞,殷海博,葛楠.基于Verilog语言的4位二进制可逆计数器的设计[J].电脑知识与技术,2019,15(09):219-220.

[2]牟晨淏,黄铁文,贺哲明,袁申,冯洪威,王颖.基于FPGA的可逆计数器的设计[J].电脑知识与技术,2019,15(07):242-243.

[3]周庆芳.一种模可变计数器的设计[J].科技展望,2016,26(05):157.

[4]谷涛,黄勇,卢晨.一种基于Verilog的任意整数分频器实现方法[J].广西民族大学学报(自然科学版),2015,21(04):62-66.

[5]蔡燕玲.基于单片机的多功能番茄钟设计[J].信息技术与信息化,2015(10):79-81.

# 总结

总结部分包括课程收获，心得体会，对综合实验的总结以及意见建议四个部分，主要写的是我对于本课程以及实验的一些感想。

## 课程收获

经过8周的学习，我们终于结束了《数字电子技术》这门课的学习，而随后的几周，我们又完成了《数字电子技术》的相关实验及其相关课程。

通过学习《数字电子技术》这门课，我学到了很多，不仅学到了充实的理论知识，为我之后进一步的专业课的学习奠定了基础，更掌握了Verilog HDL这门硬件编程语言，为我们以后的科研和学习提供了一种新的工具。

在实验过程中，我们需要将基础知识与硬件编程结合起来，在这个过程中，我们进一步提升了对基本数字电子技术知识的理解，同时，还需要查阅用户手册、数据手册，在这个过程中也极大地提高了我的自学能力。

## 心得体会

在进行实验的过程中，我发现硬件编程与软件编程有很大的区别：

首先，我们需要将程序中的变量与FPGA的引脚对应起来，在这个过程中，我必须学会查阅引脚定义然后将其与变量关联；

其次，对于Verilog HDL语言，在完成源文件的编程后，还可以通过仿真程序来验证程序逻辑的正确性，通过波形，我们可以观察到不同情况下输入输出的高低电平情况；

最后，在对FPGA编程的过程中，我明白了硬件编程不仅要能完成需求，更要考虑到芯片的占用情况，考虑程序运行的时间以及程序的可读性和可修改性等等各项因素。仅仅只是完成了功能是远远不够的，还需要尽量简化程序的逻辑，以提高运行的效率。

## 实验总结

对于番茄钟的实验，我们前几次实验完成了25分钟倒计时，数码管显示以及有限状态机的实验，通过top文件简单组合便可以实现番茄钟的基本功能，而要实现提高要求，则需要其他模块的编写。

我通过设计不同的模块，包括时间状态转换模块，时钟设定模块，多位显示模块，周期计数模块，然后将这些模块在顶层文件中连接，以实现番茄钟的全部功能，这样编写便于我后期调试，修改错误，同时，模块化的设计，更加便于我实现其他创意功能。同时，如果后期有其他项目需要进行Verilog HDL，我也可以直接使用其中的模块进行调用。

这样的思路同样可以运用到其他的硬件编程中，模块化的设计使代码更加的简洁高效，易于阅读和修改。

# 附 录

**附录部分为源程序，包括各个.v文件、约束文件。**

## 源程序

### 顶层模块top

`timescale 1ns / 1ps

module top(

input dclk\_in, //时钟信号

input reset\_in, //开关

input vauxpl,vauxnl, //模拟电压输入

input secadjin, //控制秒

input minadjin, //控制分钟

input wire key2, //按键2

input wire key1, //按键1

input wire timerstin, //清零

output wire [13:0]a\_to\_g, //数码管显示控制

output wire [7:0]anout,

input wire customtimeena //周期控制

);

wire [7:0] led,an,anmid,pwmctrl;

wire [15:0] customtime;

wire secadjout,minadjout,secadjmid,minadjmid,timerstmid,timerstout,xiaodoumaichong;

//创意功能：可以通过EGO1自带的ADC芯片控制数码管的亮度

//模数转换

XADC\_channel(

.dclk\_in(dclk\_in),

.reset\_in(1),

.vauxnl(vauxnl),

.vauxpl(vauxpl),

.led(led)

);

//设定PWM周期

function [7:0] nonlinearrevise(input [7:0] x);

nonlinearrevise = x \* x / 255;

endfunction

assign pwmctrl = nonlinearrevise(led);

genvar i;

generate

for(i = 0;i < 8;i = i + 1)

begin: ENLOOP

pwmctrl

(

.clk(dclk\_in),

.linein(an[i]),

.lineout(anmid[i]),

.pwmctrl(pwmctrl)

);

end

endgenerate

assign anout = reset\_in?anmid:8'bz;

//番茄钟基本功能

tomato\_clock(

.clk(dclk\_in),

.key1(key1),

.key2(key2 || (customtimeena && (secadjin || minadjin || timerstin))),

.a\_to\_g(a\_to\_g),

.an(an),

.reset\_in(reset\_in),

.customtime(customtime),

.customtimeena(customtimeena)

);

//产生秒脉冲

Sec\_pule\_generator #(.N(100\_000))(

.clk(dclk\_in),

.sec\_pule(xiaodoumaichong)

);

//开关消抖

ButtonEdgeDetect (

.i\_sig1(secadjin),

.i\_sig2(minadjin),

.clk(xiaodoumaichong),

.o\_sigflag1\_r(secadjmid),

.o\_sigflag2\_r(minadjmid)

);

//开关消抖

ButtonEdgeDetect (

.i\_sig1(timerstin),

.i\_sig2(0),

.clk(xiaodoumaichong),

.o\_sigflag1\_r(timerstmid)

);

//分钟显示

differenciator min(

.clk(dclk\_in),

.NumIn(minadjmid),

.NumOut(minadjout)

);

//秒显示

differenciator sec(

.clk(dclk\_in),

.NumIn(secadjmid),

.NumOut(secadjout)

);

//周期显示

differenciator timerst(

.clk(dclk\_in),

.NumIn(timerstmid),

.NumOut(timerstout)

);

//按键调节时间

pressadjuster(

.minadj(minadjout),

.secadj(secadjout),

.timerst(timerstout),

.en(reset\_in),

.clk(dclk\_in),

.numout(customtime)

);

endmodule

### 模数转换顶层模块XADC\_channel

`timescale 1ns / 1ps

//数码管亮度调节

module XADC\_channel(

input dclk\_in,

input reset\_in,

input vauxpl,vauxnl,

output [7:0]led

);

wire [15:0] do\_out;

wire drdy\_out;

wire eoc\_out;

xadc\_wiz\_0 U1 (

.di\_in(16'b0),

.daddr\_in(7'h11),

.den\_in(eoc\_out),

.dwe\_in(1'b0),

.drdy\_out(drdy\_out),

.do\_out(do\_out),

.dclk\_in(dclk\_in),

.reset\_in(~reset\_in),

.vp\_in(),

.vn\_in(),

.vauxp1(vauxpl),

.vauxn1(vauxnl),

.channel\_out(),

.eoc\_out(eoc\_out),

.alarm\_out(),

.eos\_out(),

.busy\_out()

);

read U2(

.clk(dclk\_in),

.rst(~reset\_in),

.drdy\_out(drdy\_out),

.do\_out(do\_out),

.led(led)

);

endmodule

### 模拟信号读取模块read

`timescale 1ns / 1ps

module read(

input clk,

input rst,

input [15:0]do\_out,

input drdy\_out,

output reg [7:0]led

);

always @(posedge clk or posedge rst)

if(rst)

led<=12'b0;

else if(drdy\_out==1)

led<=do\_out[15:8];

endmodule

### 番茄钟顶层模块tomato\_clock

`timescale 1ns / 1ps

//番茄钟模块

//正常番茄时间倒计时与数码管显示

module tomato\_clock(

input wire clk,

input wire key1,

input wire key2,

output wire [13:0]a\_to\_g,

output wire [7:0]an,

input wire [15:0] customtime,

input wire customtimeena,

input wire reset\_in

);

wire m1,m2, modeind, xiaodoumaichong;

//秒脉冲产生

Sec\_pule\_generator #(.N(100000))(

.clk(clk),

.sec\_pule(xiaodoumaichong)

);

//按键消抖

ButtonEdgeDetect U0(

.i\_sig1(key1),

.i\_sig2(key2),

.clk(xiaodoumaichong),

.o\_sigflag1\_r(m1),

.o\_sigflag2\_r(m2)

);

//25分钟倒计时

counter25\_top U1(

.clk(clk),

.key1(m1),

.key2(m2),

.a\_to\_g(a\_to\_g[6:0]),

.an(an[3:0]),

.modeind(modeind),

.customtime(customtime),

.customtimeena(customtimeena),

.reset\_in(reset\_in)

);

edgecounter U2(

.rst\_n(reset\_in),

.ena(!(key1||key2)),

.clk(clk),

.in(modeind),

.a\_to\_g(a\_to\_g[13:7]),

.an(an[7:6])

);

endmodule

### 秒脉冲产生模块Sec\_pule\_generator

`timescale 1ns / 1ps

// 产生秒脉冲

// 本程序为便于观察实验现象将秒脉冲调快

module Sec\_pule\_generator#(parameter N = 1000000)(

input clk,

output reg sec\_pule//输出脉冲

);

parameter M=19;

reg [M:0] cnt;

initial cnt=20'b0;//初始化计数变量

always@(posedge clk)

begin

if(cnt == N)

cnt <= 0;

else

cnt <= cnt+1;

end

always@(posedge clk)

begin

if(cnt == N)

sec\_pule <= 1;

else

sec\_pule <= 0;

end

endmodule

### 开关消抖模块ButtonEdgeDetect

`timescale 1ns / 1ps

//开关消抖

module ButtonEdgeDetect(

input i\_sig1,

input i\_sig2,

input clk,

output wire o\_sigflag1\_r,

output wire o\_sigflag2\_r

);

reg i\_sig1\_reg0;

reg i\_sig1\_reg1;

reg i\_sig2\_reg0;

reg i\_sig2\_reg1;

always@(posedge clk)

begin

i\_sig1\_reg0<=i\_sig1;

i\_sig1\_reg1<=i\_sig1\_reg0;

i\_sig2\_reg0<=i\_sig2;

i\_sig2\_reg1<=i\_sig2\_reg0;

end

assign o\_sigflag1\_r=i\_sig1\_reg0&(~i\_sig1\_reg1);

assign o\_sigflag2\_r=i\_sig2\_reg0&(~i\_sig2\_reg1);

endmodule

### 25分钟倒计时顶层模块counter25\_top

`timescale 1ns / 1ps

module counter25\_top(

input wire clk,

input wire key1,

input wire key2,

output wire [6:0]a\_to\_g,

output wire [3:0]an,

output wire modeind,

input wire [15:0] customtime,

input wire customtimeena,

input wire reset\_in

);

wire jinwei;//用于表征进位的变量

wire a;

wire b;

wire c;

//产生秒脉冲

Sec\_pule\_generator U0(

.clk(clk),

.sec\_pule(jinwei)

);

//状态转换

State U1(

.clk(clk),

.key1(key1),

.key2(key2),

.cnt\_en(a),

.load(b)

);

//倒计时模块

counter25 U2(

.clk\_sec(jinwei),

.clk(clk),

.cnt\_en(a),

.load(b),

.a\_to\_g(a\_to\_g),

.key2(key2),

.an(an),

.carry2(modeind),

.customtime(customtime),

.customtimeena(customtimeena),

.rst\_n(reset\_in)

);

endmodule

### 状态记录模块State

`timescale 1ns / 1ps

module State(

input clk,

input key1,

input key2,

output reg cnt\_en,

output reg load

);

wire rst=0;

parameter IDLE=2'b00,

S1=2'b01,

S2=2'b10;

reg [1:0]cstate;

reg [1:0]nstate;

always@(posedge clk or posedge rst)

if(rst)

cstate<=IDLE;

else

cstate<=nstate;

always@(cstate,key1,key2)

if(cstate==IDLE&&key1==1)

nstate=S1;

else if(cstate==IDLE&&key1==0)

nstate=IDLE;

else if(cstate==S1&&key1==1)

nstate=S2;

else if(cstate==S1&&key1==0)

nstate=S1;

else if(cstate==S2&&key1==1&&key2==0)

nstate=S1;

else if(cstate==S2&&key1==0&&key2==0)

nstate=S2;

else if(cstate==S2&&key1==1&&key2==1)

nstate=S2;

else if(cstate==S2&&key1==0&&key2==1)

nstate=IDLE;

else nstate=cstate;

always@\*

if(rst)

begin cnt\_en<=1'b0;load<=1'b1;end

else if(cstate==IDLE)

begin cnt\_en<=1'b0;load<=1'b1;end

else if(cstate==S1)

begin cnt\_en<=1'b1;load<=1'b0;end

else if(cstate==S2)

begin cnt\_en<=1'b0;load<=1'b0;end

else begin cnt\_en<=1'b0;load<=1'b1;end

endmodule

### 25分钟倒计时模块counter25

`timescale 1ns / 1ps

module counter25(

input wire clk,

input wire clk\_sec,

input wire rst\_n,

input wire cnt\_en,

input wire load,

input wire key2,

input wire [15:0] customtime,

input wire customtimeena,

output reg[6:0]a\_to\_g,

output reg[3:0]an,

output reg carry2

);

reg [15:0] cnt25;

wire [15:0] startnum[1:0];

reg carry;

reg innerbreak;

reg [3:0] counter = 0;

parameter [3:0] MAXNOTIF = 15;

//初始化

initial begin

cnt25[3:0]=0;

cnt25[7:4]=0;

cnt25[11:8]=5;

cnt25[15:12]=2;

carry=0;

carry2=0;

end

always@(posedge key2 || innerbreak) carry=~carry;

assign startnum[0] = customtimeena?customtime:16'h500;

assign startnum[1] = customtimeena?customtime:16'h1500;

always@(posedge clk\_sec or negedge rst\_n)

if(!rst\_n)//如果复位信号有效则输出2500

begin

cnt25[3:0]=0;

cnt25[7:4]=0;

cnt25[11:8]=5;

cnt25[15:12]=2;

carry2=0;

end

else

begin

innerbreak = 0;

if(cnt\_en==0&&carry != carry2)

begin

if(carry==0)

begin

cnt25 <= startnum[1];

carry2<=0;

end

else if(carry==1)

begin

cnt25 <= startnum[0];

carry2<=1;

end

end

else if(cnt\_en==1&&load==0)

begin

if(carry2==0)

begin

if(cnt25[15:12]==0&&cnt25[11:8]==0&&cnt25[7:4]==0&&cnt25[3:0]==0||cnt25 == 16'hFFFF||cnt25 == 16'hAAAA)

begin

if(counter != MAXNOTIF)

begin

counter = counter + 1;

if(cnt25 == 16'hAAAA) cnt25 <= 16'hFFFF;

else cnt25 <= 16'hAAAA;

end

else

begin

cnt25 <= startnum[0];

carry2<=1;

innerbreak = 1;

counter <= 0;

end

end

else

begin

cnt25[3:0]<=cnt25[3:0]-1;

if(cnt25[3:0]==0)

begin

cnt25[3:0]<=9;

cnt25[7:4]<=cnt25[7:4]-1;

end

if(cnt25[7:4]==0&&cnt25[3:0]==0)

begin

cnt25[7:4]<=5;

cnt25[11:8]<=cnt25[11:8]-1;

end

if(cnt25[11:8]==0&&cnt25[7:4]==0&&cnt25[3:0]==0)

begin

cnt25[11:8]<=9;

cnt25[15:12]<=cnt25[15:12]-1;

end

end

end

else if (carry2==1)

begin

if(cnt25[15:12]==0&&cnt25[11:8]==0&&cnt25[7:4]==0&&cnt25[3:0]==0||cnt25 == 16'hFFFF||cnt25 == 16'hAAAA)

if(counter != MAXNOTIF)

begin

counter = counter + 1;

if(cnt25 == 16'hAAAA)

cnt25 <= 16'hFFFF;

else

cnt25 <= 16'hAAAA;

end

else

begin

cnt25 <= startnum[1];

carry2<=0;

counter <= 0;

innerbreak = 1;

end

else

begin

cnt25[3:0]<=cnt25[3:0]-1;

if(cnt25[3:0]==0)

begin

cnt25[3:0]<=9;

cnt25[7:4]<=cnt25[7:4]-1;

end

if(cnt25[7:4]==0&&cnt25[3:0]==0)

begin

cnt25[7:4]<=5;

cnt25[11:8]<=cnt25[11:8]-1;

end

if(cnt25[11:8]==0&&cnt25[7:4]==0&&cnt25[3:0]==0&&cnt25[15:12]!=0)

begin

cnt25[11:8]<=9;

cnt25[15:12]<=cnt25[15:12]-1;

end

end

end

end

end

wire [15:0]x;

assign x[3:0]=cnt25[3:0];

assign x[7:4]=cnt25[7:4];

assign x[11:8]=cnt25[11:8];

assign x[15:12]=cnt25[15:12];

reg[18:0]clkdiv;

always @(posedge clk or negedge rst\_n)begin

if(!rst\_n)

clkdiv<=19'd0;

else

clkdiv<=clkdiv+1;

end

wire [1:0]bitcnt;

assign bitcnt=clkdiv[18:17];

always @\*

begin

if(!rst\_n)

an=4'd0;

else

an=4'd0;

an[bitcnt]=1;

end

reg [3:0]digit;

always@(\*)

begin

if(!rst\_n)

digit=4'd0;

else

case(bitcnt)

2'd0:digit=x[3:0];

2'd1:digit=x[7:4];

2'd2:digit=x[11:8];

2'd3:digit=x[15:12];

default:digit=4'd0;

endcase

end

//数码管显示

always @(\*) begin

if(!rst\_n)

a\_to\_g=7'b1111111;

else

case(digit)

0:a\_to\_g=7'b1111110;

1:a\_to\_g=7'b0110000;

2:a\_to\_g=7'b1101101;

3:a\_to\_g=7'b1111001;

4:a\_to\_g=7'b0110011;

5:a\_to\_g=7'b1011011;

6:a\_to\_g=7'b1011111;

7:a\_to\_g=7'b1110000;

8:a\_to\_g=7'b1111111;

9:a\_to\_g=7'b1111011;

10:a\_to\_g=7'b1111111;

4'b1111:a\_to\_g=7'b0000000;

default a\_to\_g=7'b1111110;

endcase

end

endmodule

### 周期计数信号edgecounter

`timescale 1ns / 1ps

module edgecounter(

input rst\_n,

input ena,

input clk,

input in,

output reg [6:0] a\_to\_g,

output reg [1:0] an

);

reg [7:0] count = 0;

reg [7:0] x = 0;

always @(negedge in or negedge rst\_n)

if(!rst\_n)

count <= 0;

else if(ena)

count = count + 1;

function [7:0] binary2bcd(input [7:0] b);

begin

binary2bcd[3:0] = b % 10;

binary2bcd[7:4] = b / 10;

end

endfunction

always@(count)

x = binary2bcd(count);

reg[17:0]clkdiv;

always @(posedge clk or negedge rst\_n)

begin

if(!rst\_n)

clkdiv<=19'd0;

else

clkdiv<=clkdiv+1;

end

wire bitcnt;

assign bitcnt=clkdiv[17];

always @\*

begin

if(!rst\_n)

an=2'd0;

else

an=2'd0;

an[bitcnt]=1;

end

reg [3:0]digit;

always@(\*)

begin

if(!rst\_n)

digit=4'd0;

else

case(bitcnt)

2'd0:digit=x[3:0];

2'd1:digit=x[7:4];

2'd2:digit=x[11:8];

2'd3:digit=x[15:12];

default:digit=4'd0;

endcase

end

//数码管显示

always @(\*)

if(!rst\_n)

a\_to\_g=7'b1111111;

else

case(digit)

0:a\_to\_g=7'b1111110;

1:a\_to\_g=7'b0110000;

2:a\_to\_g=7'b1101101;

3:a\_to\_g=7'b1111001;

4:a\_to\_g=7'b0110011;

5:a\_to\_g=7'b1011011;

6:a\_to\_g=7'b1011111;

7:a\_to\_g=7'b1110000;

8:a\_to\_g=7'b1111111;

9:a\_to\_g=7'b1111011;

10:a\_to\_g=7'b1111111;

4'b1111:a\_to\_g=7'b0000000;

default a\_to\_g=7'b1111110;

endcase

endmodule

### 微分器differenciator

`timescale 1ns / 1ps

module differenciator(

input clk,

input NumIn,

input NumOut

);

reg cstate = 0, nstate = 0;

always @(posedge clk)

cstate = nstate;

always @(posedge clk)

nstate = NumIn;

assign NumOut = (cstate == 0) && (nstate == 1)?1:0;

endmodule

### 按键控制周期pressadjuster

`timescale 1ns / 1ps

module pressadjuster(

input minadj,

input secadj,

input timerst,

input en,

input clk,

output wire[15:0] numout

);

reg [6:0] min = 0, sec = 0;

function [7:0] binary2bcd(input [7:0] b);

begin

binary2bcd[3:0] = b % 10;

binary2bcd[7:4] = b / 10;

end

endfunction

assign numout = {binary2bcd(min), binary2bcd(sec)};

//调节番茄钟的周期

always@(posedge clk)

if(en)

if(timerst)

if(min == 0 && sec == 0)

min = 25;

else

begin

min = 0;

sec = 0;

end

else if(minadj)//按键有效分钟数加1

if(min > 99) //分钟上限99

min = 0;

else

min = min + 1;

else if(secadj) //按键有效秒数加1

if(min > 99)

sec = 0;

else

sec = sec + 1;

endmodule

### PWM控制器pwmctrl

`timescale 1ns / 1ps

module pwmctrl

(

input wire clk,

input linein,

output lineout,

input [7:0] pwmctrl

);

reg [7:0]count = 0;

always@(posedge clk)

count = count + 1;

assign lineout = count < pwmctrl?linein:1'bz;

endmodule

## 约束文件

### CON\_of\_tomato

set\_property PACKAGE\_PIN B4 [get\_ports {a\_to\_g[13]}]

set\_property PACKAGE\_PIN A4 [get\_ports {a\_to\_g[12]}]

set\_property PACKAGE\_PIN A3 [get\_ports {a\_to\_g[11]}]

set\_property PACKAGE\_PIN B1 [get\_ports {a\_to\_g[10]}]

set\_property PACKAGE\_PIN A1 [get\_ports {a\_to\_g[9]}]

set\_property PACKAGE\_PIN B3 [get\_ports {a\_to\_g[8]}]

set\_property PACKAGE\_PIN B2 [get\_ports {a\_to\_g[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[13]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[12]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[11]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[10]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[9]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[7]}]

set\_property PACKAGE\_PIN G2 [get\_ports {anout[7]}]

set\_property PACKAGE\_PIN C2 [get\_ports {anout[6]}]

set\_property PACKAGE\_PIN C1 [get\_ports {anout[5]}]

set\_property PACKAGE\_PIN H1 [get\_ports {anout[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[4]}]

set\_property PACKAGE\_PIN D4 [get\_ports {a\_to\_g[6]}]

set\_property PACKAGE\_PIN E3 [get\_ports {a\_to\_g[5]}]

set\_property PACKAGE\_PIN D3 [get\_ports {a\_to\_g[4]}]

set\_property PACKAGE\_PIN F4 [get\_ports {a\_to\_g[3]}]

set\_property PACKAGE\_PIN F3 [get\_ports {a\_to\_g[2]}]

set\_property PACKAGE\_PIN E2 [get\_ports {a\_to\_g[1]}]

set\_property PACKAGE\_PIN D2 [get\_ports {a\_to\_g[0]}]

set\_property PACKAGE\_PIN G1 [get\_ports {anout[3]}]

set\_property PACKAGE\_PIN F1 [get\_ports {anout[2]}]

set\_property PACKAGE\_PIN E1 [get\_ports {anout[1]}]

set\_property PACKAGE\_PIN G6 [get\_ports {anout[0]}]

set\_property PACKAGE\_PIN V1 [get\_ports key1]

set\_property PACKAGE\_PIN R11 [get\_ports key2]

set\_property IOSTANDARD LVCMOS33 [get\_ports key2]

set\_property IOSTANDARD LVCMOS33 [get\_ports key1]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {anout[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a\_to\_g[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[0]}]

set\_property PACKAGE\_PIN F6 [get\_ports {led[7]}]

set\_property PACKAGE\_PIN G4 [get\_ports {led[6]}]

set\_property PACKAGE\_PIN G3 [get\_ports {led[5]}]

set\_property PACKAGE\_PIN J4 [get\_ports {led[4]}]

set\_property PACKAGE\_PIN H4 [get\_ports {led[3]}]

set\_property PACKAGE\_PIN J3 [get\_ports {led[2]}]

set\_property PACKAGE\_PIN J2 [get\_ports {led[1]}]

set\_property PACKAGE\_PIN K2 [get\_ports {led[0]}]

set\_property PACKAGE\_PIN P17 [get\_ports dclk\_in]

set\_property IOSTANDARD LVCMOS33 [get\_ports dclk\_in]

set\_property PACKAGE\_PIN R1 [get\_ports reset\_in]

set\_property PACKAGE\_PIN C12 [get\_ports vauxpl]

set\_property IOSTANDARD LVCMOS33 [get\_ports vauxnl]

set\_property IOSTANDARD LVCMOS33 [get\_ports vauxpl]

set\_property IOSTANDARD LVCMOS33 [get\_ports reset\_in]

set\_property PACKAGE\_PIN N4 [get\_ports {customtimeena}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {customtimeena}]

set\_property PACKAGE\_PIN U4 [get\_ports {minadjin}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {minadjin}]

set\_property PACKAGE\_PIN R17 [get\_ports {secadjin}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {secadjin}]

set\_property PACKAGE\_PIN R15 [get\_ports {timerstin}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {timerstin}]