西安交通大学

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

1.1 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。

1.2 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。

1.3 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。

1.4 设计难点与亮点

1.4.1 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。

1.4.2 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。

2　系统设计

2.1 总体结构及功能设计

2.1.1 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。

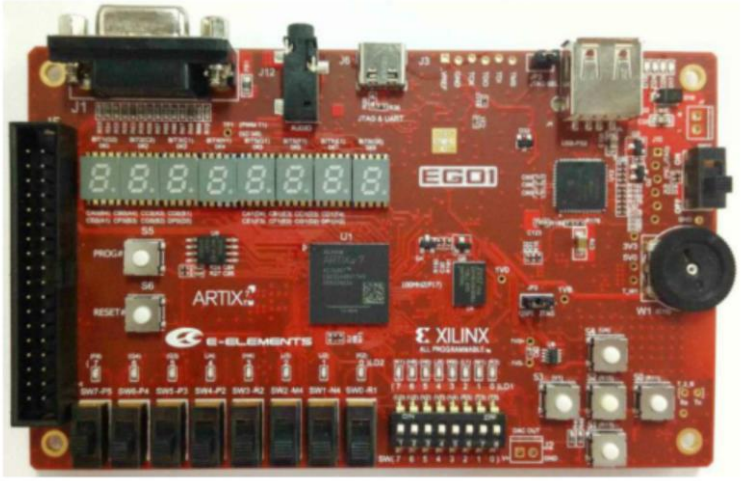

图 2.1 EGO1 开发板

2.1.2 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。

2.1.3 输入输出设计

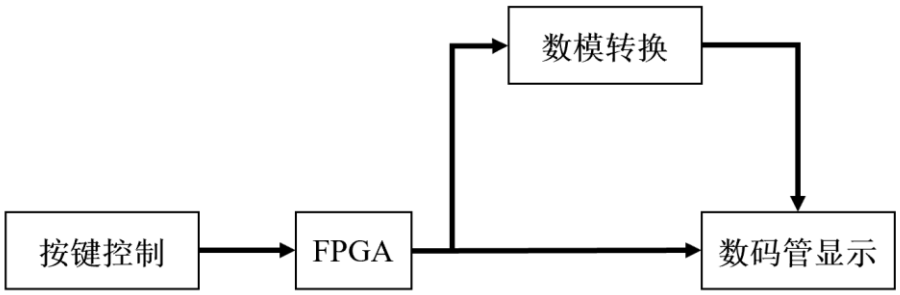

图 2.2 系统原理框图

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。

2.2 子模块设计

2.2.1 子模块划分

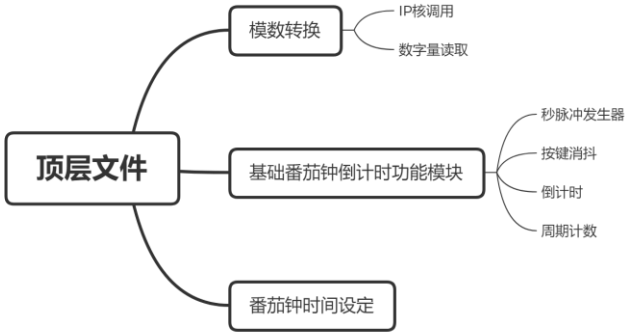

图 2.3 子模块划分示意图

2.2.2 RTL 顶层逻辑图

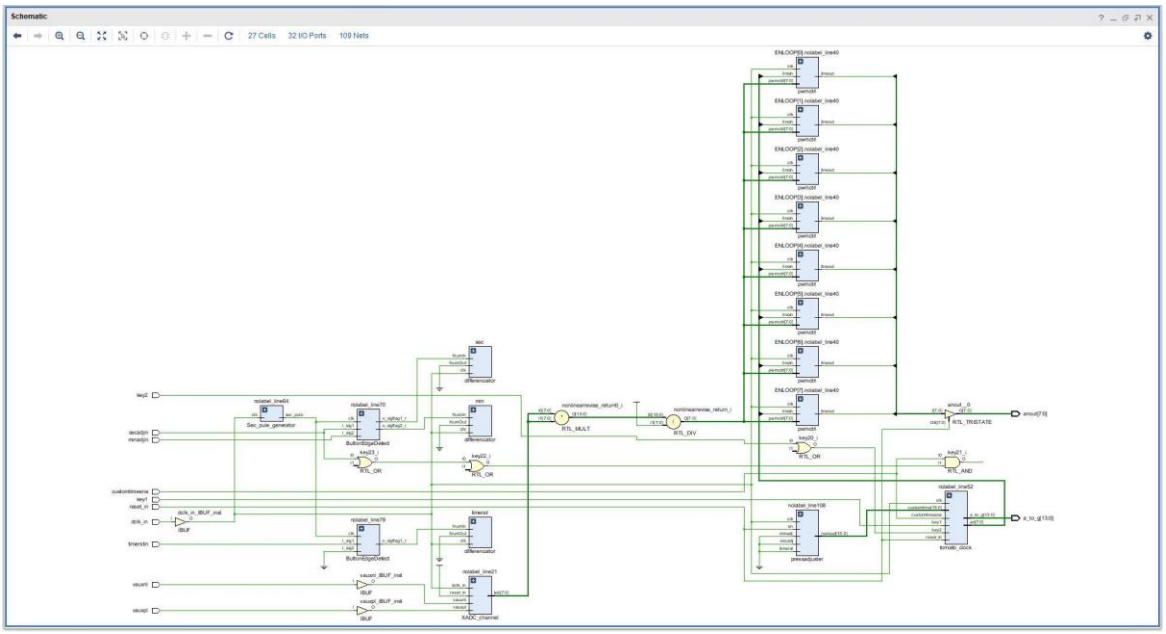

图 2.6 RTL 分析顶层逻辑图

2.2.3 子模块设计思路

（1）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。

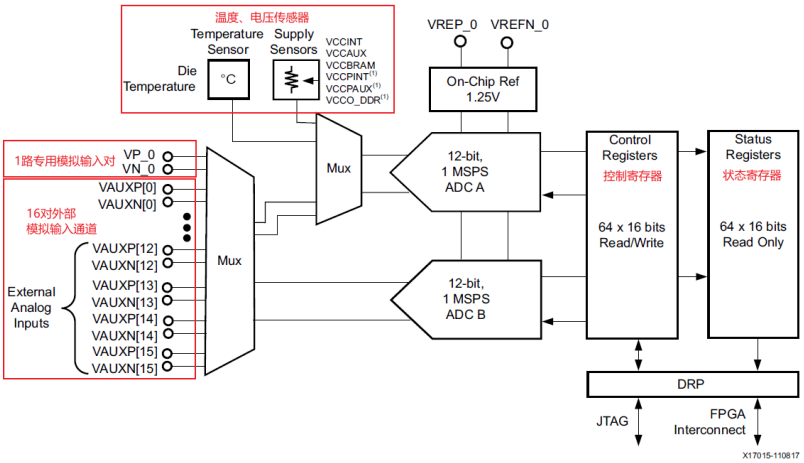

Figure 1-1: **XADC Block Diagram**

图 2.4 XDAC 原理

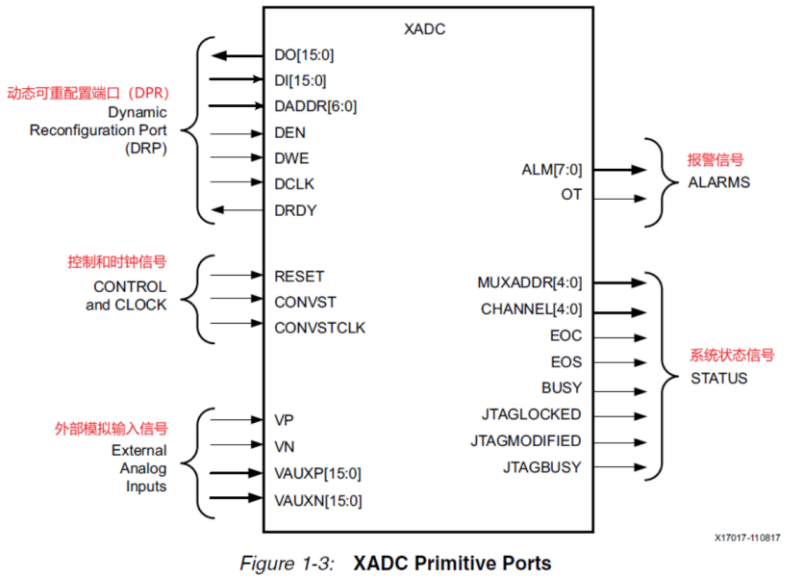

Figure 1-3: **XADC Primitive Ports**

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。

（2）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。

（3）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。

2.3 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。

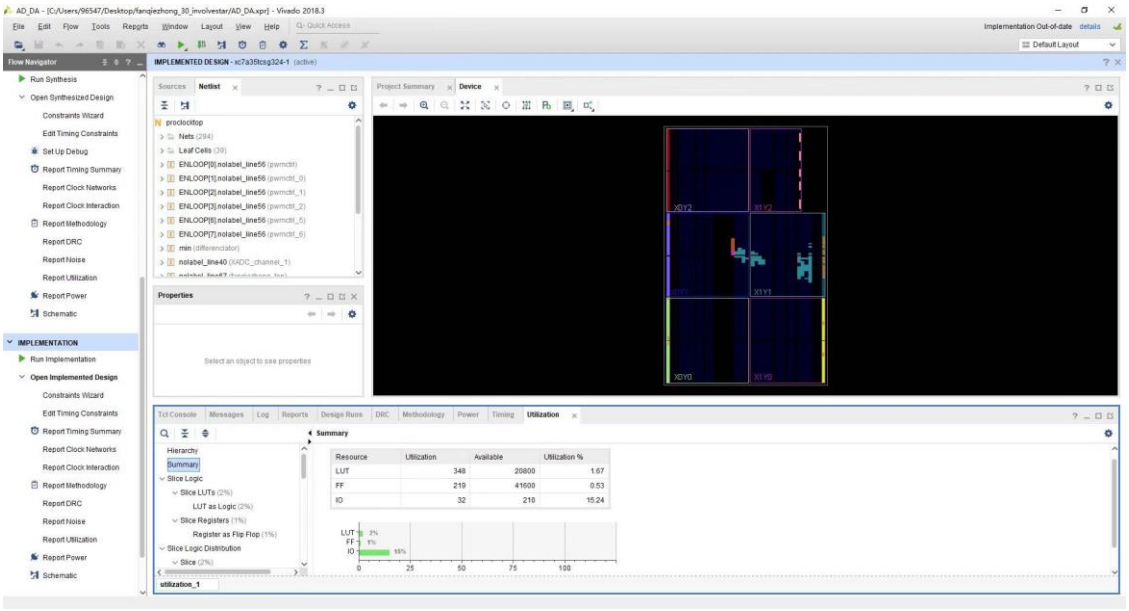

图 2.7 Imlementation report

3　系统调试

3.1 遇到的问题

3.1.1 倒计时功能发现问题

（1）初值显示不正常

在 25 分钟倒计时状态中，初值显示不正常。

（2）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。

3.1.2 其他问题

（1）查看 RTL 级原理图时报错

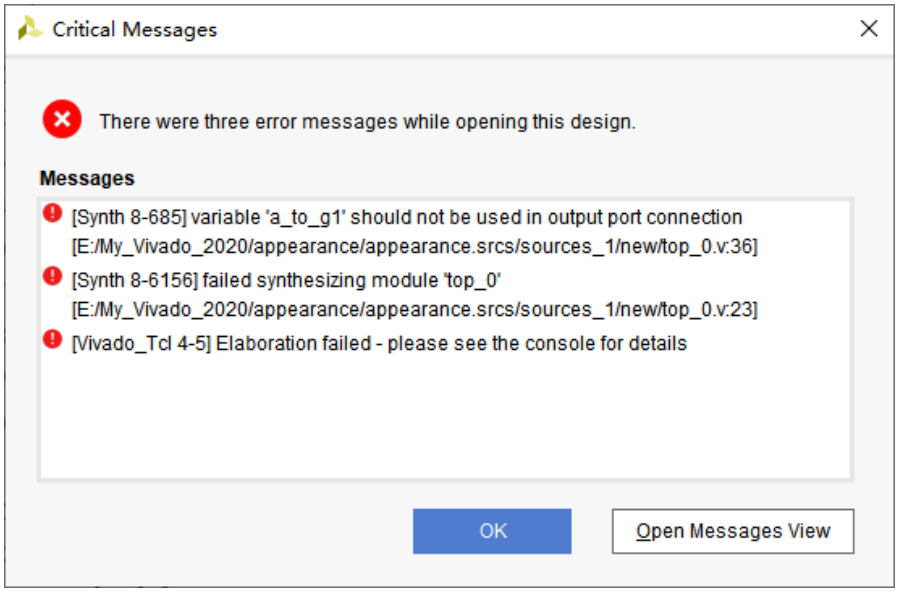

图 3.1 报错信息

（2）板子无法连接

hardware 窗口显示 localhost(0)

（3）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

3.2 解决方案

3.2.1 倒计时功能问题解决

（1）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。

（2）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。

3.2.2 其他问题解决

（1）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。

（2）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。

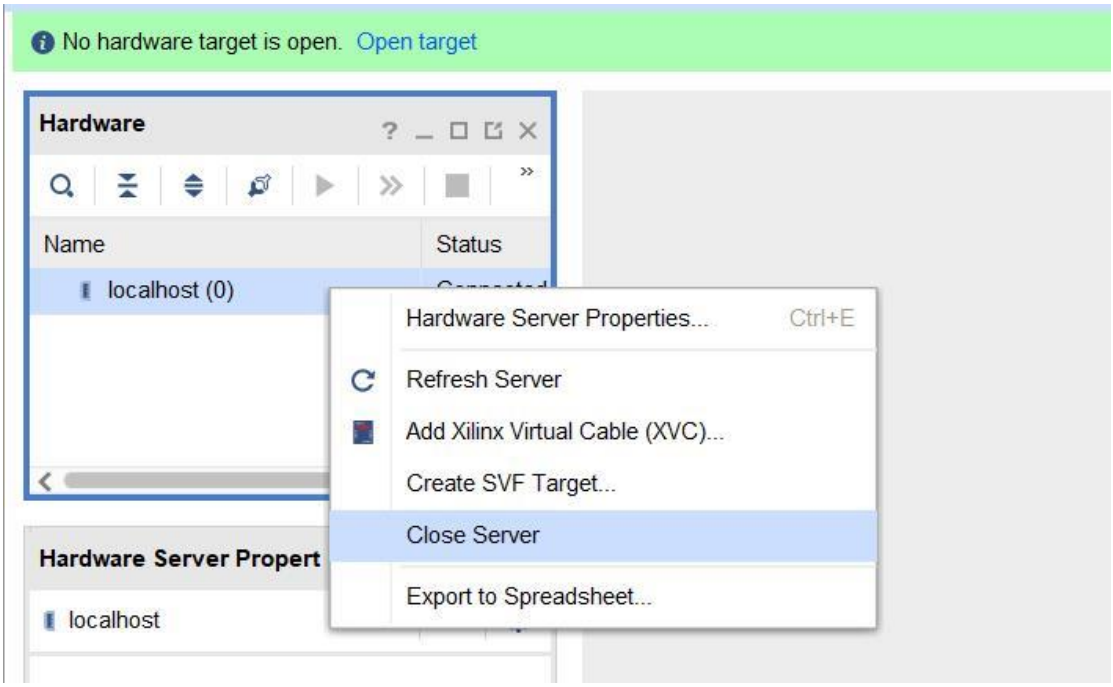

图 3.2 解决方法

（3）整合时报错

assign reg 型变量，reg 型变量不能使用 assign 命令。

（4）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

3.3 设计改进

3.3.1 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。

4　系统使用说明

4.1 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。

4.2 按键控制

4.2.1 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。

4.2.2 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。

4.3 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。

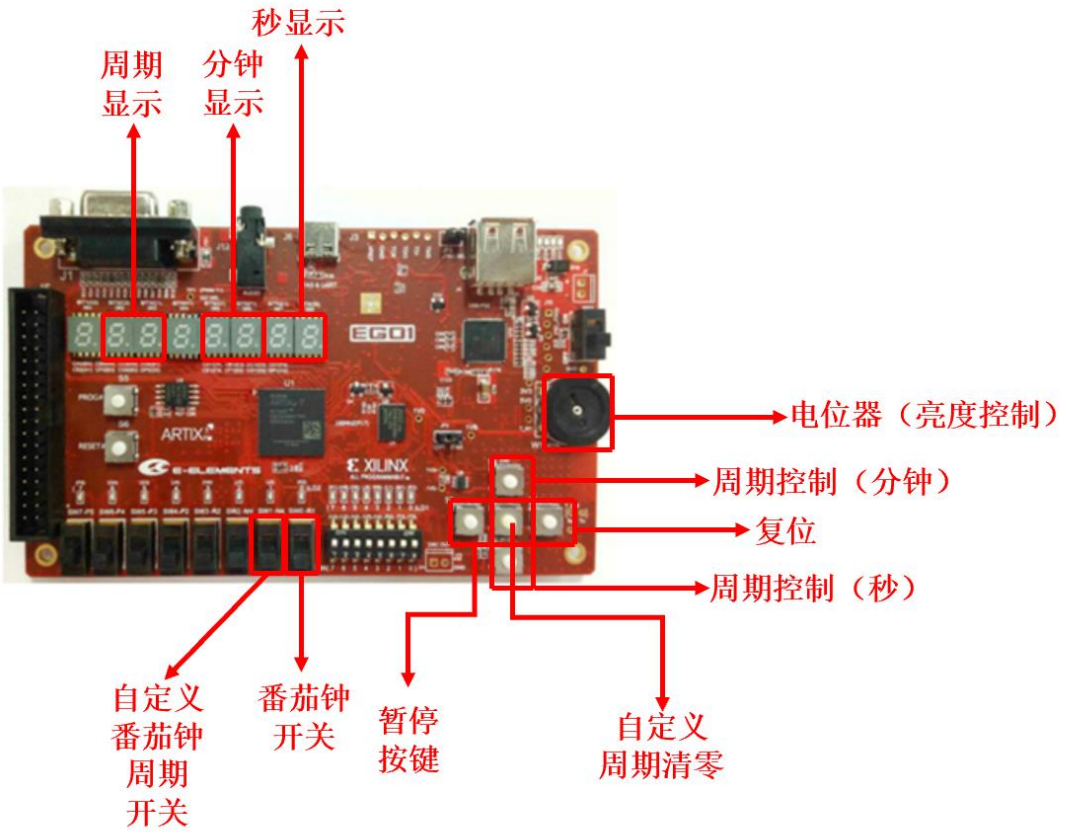

图 4.1　番茄钟功能按键示意图

5 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.

6 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。

6.1 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。

6.2 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。

6.3 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。

7 附 录

附录部分为源程序，包括各个.v 文件、约束文件。

7.1 源程序

7.1.1 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
```

7.1.2 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 vauxp1,vauxn1,
    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
```

7.1.3 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
```

7.1.4 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
```

7.1.5 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
```

7.1.6 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
```

7.1.7 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
```

7.1.8 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
```

7.1.9 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
```

7.1.10 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
```

7.1.11 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
```

7.1.12 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
```

7.1.13 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
```

7.2 约束文件

7.2.1 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}]
```