2019-2020学年度第一学期

电子技术基础课程设计

--基于FPGA的智能家居设计



院 系： 电子信息与通信学院

专 业： 通信工程

班 级： 通信1704

姓 名：

学 号： U201713480

指导教师： 潘晓明

目录

[课设报告创作说明 4](#_Toc57146876)

[一、 硬件课设概述 4](#_Toc57146877)

[1.1课设目的 4](#_Toc57146878)

[1.2整体描述 4](#_Toc57146879)

[1.3项目要求 5](#_Toc57146880)

[1.3.1基本要求： 5](#_Toc57146881)

[1.3.2进阶要求： 5](#_Toc57146882)

[1.4参考资料 5](#_Toc57146883)

[1.5开发环境及所用设备 6](#_Toc57146884)

[二、 项目背景及需求分析 6](#_Toc57146885)

[2.1背景概述 6](#_Toc57146886)

[2.2发展现状 8](#_Toc57146887)

[2.3 开发软硬件介绍 9](#_Toc57146888)

[2.3.1NEXYS4： 9](#_Toc57146889)

[2.3.2Vivado： 10](#_Toc57146890)

[2.4需求分析 11](#_Toc57146891)

[三、 系统综述 12](#_Toc57146892)

[3.1系统设计 12](#_Toc57146893)

[3.2结构框图 13](#_Toc57146894)

[四、 课设历程 13](#_Toc57146895)

[4.1 DHT11介绍 13](#_Toc57146896)

[４.２DHT１１数据采集过程抽象 16](#_Toc57146897)

[４.３温湿度测量代码实现 18](#_Toc57146898)

[4.4学习过程回顾 20](#_Toc57146899)

[4.4.1资料整理 20](#_Toc57146900)

[4.4.2曲折的学习过程 21](#_Toc57146901)

[４.４.３回顾分析 22](#_Toc57146902)

[五、 综合调试结果 22](#_Toc57146903)

[５.１　DHT１１温湿度测量模块调试 22](#_Toc57146904)

[５.２音乐播放模块调试 23](#_Toc57146905)

[５.３预设参数修改模块调试 24](#_Toc57146906)

[５.４整机系统调试 24](#_Toc57146907)

[六、 课设总结 25](#_Toc57146908)

[６.１课设分工 25](#_Toc57146909)

[6.2部分遇到的问题及解决方案 25](#_Toc57146910)

[6.3未来展望 26](#_Toc57146911)

[6.4心得体会 26](#_Toc57146912)

[七、附录代码 26](#_Toc57146913)

[７.１顶层模块 26](#_Toc57146914)

[７.２DHT１１温湿度测量模块 28](#_Toc57146915)

[７.３报警模块 38](#_Toc57146916)

[７.４音乐播放模块 39](#_Toc57146917)

[７.５增减模块和选择显示模块 48](#_Toc57146918)

[７.６数码管显示模块 50](#_Toc57146919)

# 课设报告创作说明

本人郑重声明：所呈交的报告由本人撰写，报告内容依据本人参与的课程设计工作，无抄袭行为。

作者签名：

日期： 年 月 日

# 硬件课设概述

## 1.1课设目的

硬件课设的硬件课程设计的目的在于使学生通过硬件课程设计教学环节较系统地完成电子系统设计从选题、方案论证、电路设计、电路实现、装配调试、系统测试、总结报告等基本过程，加深对模拟电路、数字逻辑电路、通信电子电路、微机原理等相关课程理论知识的分析理解。引导学生把原理分析与工程设计实现相结合，掌握实际电子与通信系统设计的基本方法和一般规则，提高综合应用所学理论知识的能力，培养学生的创新思维和实践能力，为后续专业课程的学习打下坚固的基础。

## 1.2整体描述

设计并实现一个可以自动测量物理环境温湿度，根据预设的报警温度可以提供报警功能，同时可以播放音乐，并能够提供进行一些控制和数据修改功能的智能家居系统。

## 1.3项目要求

## 1.3.1基本要求：

（1）能够测量环境中的温湿度。

（2）将测量得到的数据通过FPGA上的数码管显示出来。

（3）可以在代码中设置报警温湿度，当测量得到的温湿度达到门限时触发报警机制。

## 1.3.2进阶要求：

（1）可以对测量环境温湿度等过程进行开关控制。

（2）通过开关可以控制在数码管上显示的数据是预设报警温湿度还是实际测量得到的温湿度。

（3）通过FPGA上的开关可以实现对预设报警门限的修改。

（4）增加音乐播放功能，在未报警的状态下播放背景音乐。

## 1.4参考资料

1. 谢自美，《电子线路综合设计》，华中科技大学出版社，2006
2. 康华光，《电子技术基础：模拟部分（第五版）》，高等教育出版社，2006
3. 康华光，《电子技术基础：数字部分（第五版）》，高等教育出版社，2006
4. 罗杰，《电子线路设计.实验.测试》，电子工业出版社，2008
5. <https://blog.csdn.net/qq_34952376/article/details/81193938>
6. <https://blog.csdn.net/ninquelote/article/details/105824323>
7. <https://blog.csdn.net/freshwaterpond/article/details/104651701/>
8. https://blog.csdn.net/qq\_41768748/article/details/95599779
9. <https://github.com/MaxwellLin3104/Verilog-based-Bluetooth-player-for-musical-notes>
10. https://china.xilinx.com/support/university/boards-portfolio/xup-boards/DigilentNexys4Board.html

## 1.5开发环境及所用设备

Vivado；Windows10；DHT11模块、NEXYS4、耳机、电风扇。

# 项目背景及需求分析

## 2.1背景概述

智能家居是在互联网影响之下物联化的体现。智能家居通过物联网技术将家中的各种设备（如音视频设备、[照明系统](https://baike.baidu.com/item/%E7%85%A7%E6%98%8E%E7%B3%BB%E7%BB%9F" \t "_blank)、窗帘控制、空调控制、[安防系统](https://baike.baidu.com/item/%E5%AE%89%E9%98%B2%E7%B3%BB%E7%BB%9F)、数字影院系统、[影音服务器](https://baike.baidu.com/item/%E5%BD%B1%E9%9F%B3%E6%9C%8D%E5%8A%A1%E5%99%A8)、[影柜](https://baike.baidu.com/item/%E5%BD%B1%E6%9F%9C)系统、网络家电等）连接到一起，提供家电控制、照明控制、电话远程控制、室内外遥控、防盗报警、[环境监测](https://baike.baidu.com/item/%E7%8E%AF%E5%A2%83%E7%9B%91%E6%B5%8B" \t "_blank)、暖通控制、红外转发以及可编程定时控制等多种功能和手段。与普通家居相比，智能家居不仅具有传统的居住功能，兼备建筑、网络通信、信息家电、设备自动化，提供全方位的信息交互功能，甚至为各种能源费用节约资金。

智能家居的概念起源很早，但一直未有具体的建筑案例出现，直到1984年美国联合科技公司（United Technologies Building System）将建筑设备信息化、整合化概念应用于美国康涅狄格州（Connecticut)哈特佛市（Hartford）的CityPlaceBuilding时，才出现了首栋的“智能型建筑”，从此揭开了全世界争相建造智能[家居派](https://baike.baidu.com/item/%E5%AE%B6%E5%B1%85%E6%B4%BE" \t "_blank)的序幕。

 智能家居（英文：smart home， home automation）是以住宅为平台，利用综合布线技术、网络通信技术、 安全防范技术、自动控制技术、音视频技术、人工智能技术将家居生活有关的设施集成，构建高效的住宅设施与家庭日程事务的管理系统，营造个性化的智慧健康场景，提升家居安全性、便利性、舒适性、艺术性，并实现环保节能的居住环境。  
     智能家居新定义有两个变化：增加了人工智能技术、明确了营造个性化的智慧健康场景这一功能需求。  
     人工智能（Artificial Intelligence）技术在智能家居领域的应用体现在两个方面，其一是带来了新的交互方式，包括语音识别、语音理解、动作识别、甚至脑电波识别等技术让人更加自然、灵活多样、轻松地与智能家居系统进行交互并达到预期效果，新的交互形式的最终形态会以家庭服务机器人这样的人形载体来实施与人的交互；其二是机器学习和深度学习，让人工智能能够更好地了解空间环境和人，进而主动思考判断，真正像一个管家一样为家庭成员提供服务。

   智慧健康场景不仅让产品与服务的营销过程变得有趣，更让用户愿意投入情感和创新力去参与产品设计与改进、完善，甚至用户的行动及思想成为智慧场景的一部分，成为产品服务中不可或缺的灵魂，因此，我们有必要强调用户对智慧健康场景的功能需求。

## 2.2发展现状

国外智能家居的发展概况。国外智能家居发展比较早,发展也比较迅速。早在1995年,美国和新加坡就已经大量推广智能家居,其平均安装一家智能家居消费高达7000美元,但是使用率仅占0.3%。从目前来看,美国是全世界智能家居使用最多的国家,其次是日本、德国等国家。随着智能家居的推进,美国的智能家居市场也在不断地增长和扩大,2016年,美国的家居市场容量已经达到97亿美元,并且以平均每年30亿美元左右的增长速度迅速增长,市场的总收入已经达到9912万美元,家庭普及率为5.3%左右。以现在的发展态势估计,到达2020年,北美地区的智能家居家庭普及率将达17.23%。

国内智能家居的发展概况。相较于国外的智能家居,国内的智能家居发展比较缓慢,主要分为四个时期。1994年—1999年,智能家居还处于萌芽时期,国内关于其的概念比较少,局势不太明朗。2000年—2005年,处于开创期的智能家居已经在国内得到了重视,在长三角和珠三角已经有企业开始研究智能家居并希望得到应用,此时国外的智能家居技术还没有引入国内。2006—2010年,是国内智能家居的徘徊期。由于之前国内智能家居处于摸索阶段,很多企业夸大其智能家居的特点和作用,导致宣传效果和使用效果相差较多,一部分国内的智能家居企业在这个阶段倒闭,但仍然有一部分国内企业熬过了徘徊期。此时,也有很多国外的企业正在国内布局增加其规模。到了2014年,智能家居迎来了爆发期,各种智能家居的需求和认可度都逐年增加,市场规模和销售额呈现指数型增长。目前国内做的比较好的企业有家电企业海尔提出的U-HOME,同时还有小米。虽然小米的起步较晚,但是由于其掌握手机、电视、路由器等设备,最近几年依旧发展迅猛。

智能家居产品丰富，在多个不同场景中提供更为舒适便捷、节能的人性化居住环境。智能家居的不同产品互联互通组成不同系统，逐渐从手机控制转向语音控制，到后面的自主学习自动控制等更为自然的人机交互模式。

目前，智能家居的控制多为手机，触摸屏等设备。并且每一个厂商的控制APP都不一样。这种模式的问题在于不同厂商之间的产品是不兼容的，用户的选择权利则大大降低。而随着智能家居平台的建设和不断完善，不同厂商之间的产品也可打通，并且，作为目前智能终端的手机也会慢慢向其他更适宜的智能家居产品所替代。

## 2.3 开发软硬件介绍

## 2.3.1NEXYS4：

[Nexys4](http://www.eepw.com.cn/news/listbylabel/label/Nexys4)板载大容量的FPGA芯片，拥有丰富的外部存储器资源。其板载外设还包括高速USB接口、以太网接口等，所以它可以适用于各式各样的数字系统的设计，从基础的组合逻辑电路到复杂的嵌入式处理器，都可以用[Nexys4](http://www.eepw.com.cn/news/listbylabel/label/Nexys4)来设计。另外，[Nexys4](http://www.eepw.com.cn/news/listbylabel/label/Nexys4)拥有丰富的I/O器件，还内置了加速度计、温度传感器，和带数字麦克风、扬声器的音频模块等设备，这就意味着您可以不需要其他辅助器件，单独使用Nexys4就能构建各种各样的设计。

NEXYS4是Xilinx公司开发的FPGA工作板，它具有以下资源：

1 Xilinx Artix-7 FPGA XC7A100T-1CSG324C

2 15,850个逻辑片，每片有4个6-input LUTs和8个触发器

3 4,860Kbits的快速随机存储器

4 6个时钟管理模块，每个有锁相回路（PLL）

5 240个DSP片

6 内部时钟速度超过450MHz

7 片上模数转换器（XADC）

8 128 MiB DDR2

## 2.3.2Vivado：

Vivado设计套件，是FPGA厂商赛灵思公司2012年发布的集成设计环境。包括高度集成的设计环境和新一代从系统到IC级的工具，这些均建立在共享的可扩展数据模型和通用调试环境基础上。这也是一个基于AMBA AXI4 互联规范、IP-XACT IP封装元数据、工具命令语言(TCL)、Synopsys 系统约束(SDC) 以及其它有助于根据客户需求量身定制设计流程并符合业界标准的开放式环境。赛灵思构建的的Vivado工具把各类可编程技术结合在一起，能够扩展多达1 亿个等效[ASIC](https://baike.baidu.com/item/ASIC/2014676" \t "_blank) 门的设计。

*专注于集成的组件——*为了解决集成的瓶颈问题，Vivado 设计套件采用了用于快速综合和验证C 语言算法IP 的ESL 设计，实现重用的标准算法和RTL IP 封装技术，标准IP 封装和各类系统构建模块的系统集成，模块和系统验证的仿真速度提高了3 倍，与此同时，硬件协仿真性能提升了100倍。

*专注于实现的组件——*为了解决实现的瓶颈，Vivado工具采用层次化器件编辑器和布局规划器、速度提升了3 至15 倍，且为SystemVerilog 提供了业界最好支持的逻辑综合工具、速度提升4 倍且确定性更高的布局布线引擎，以及通过分析技术可最小化时序、线长、路由拥堵等多个变量的“成本”函数。此外，增量式流程能让工程变更通知单(ECO) 的任何修改只需对设计的一小部分进行重新实现就能快速处理，同时确保性能不受影响。最后，Vivado 工具通过利用最新共享的可扩展数据模型，能够估算设计流程各个阶段的功耗、时序和占用面积，从而达到预先分析，进而优化自动化时钟门等集成功能。

## 2.4需求分析

随着物联网、大数据、移动设备普及，全新的交互技术给家居智能化带来爆发点。全球智能家居市场规模到2017年有望达到近千亿美元，年复合增长率为20%左右。智能设备的数量，预测在2020年增长到2.23亿台。以上的市场预测还仅仅是针对大型产品，比如智能冰箱、洗衣机、大型餐饮设备等。对于小型的设备，比如吸尘器、微波炉、以及很多未来可能出现的产品都还没有关注。因此智能家居行业的实际增长将很可能远远超过这个速率。2020年，所有的智能产品销售或将可能达到7亿台。随着2016年宽带世界论坛(BroadbandWorldForum)的召开，宽带运营商开始评估智能家居市场新的机遇。智能家居应用系列为运营商打造第五重播放(5thPlay)带来的市场机遇。2020年，消费者将每年在智能家居业务上花费近1300亿美元;我们预计，47%或超过600亿美元会被运营商收入囊中。全球范围内，安全和防护类应用可为运营商带来超过260亿美元的新收益。着重于为消费者提供“平和心态”的自我监测应用的全球收益将会在2020年超过140亿美元。2020年，美国将会成为规模最大的单个国家市场，其市场规模将达210亿美元;西欧的市场规模将会达到近100亿美元。在这样的背景下，我们所设计的智能家居是对未来的一次探索，具有深远的意义。

# 系统综述

## 3.1系统设计

我们想基于FPGA通过外接模块完成我们上述的功能，首先通过DHT11模块实时测量物理环境中的温湿度，通过PMOD接口和FPGA进行连接，因为DHT11是单总线结构，控制信号和数据是在同一条线上传输，所以需要通过FPGA进行精细控制，得到测量的数据，然后对数据进行相应的处理，转换为可以相对应的BCD码，在数码管上显示，整个测量过程可以通过开关进行多样控制，如：测量暂停、终止显示等。此外，在系统中我们预设有25°的报警温度，通过开关可以选择数码管显示的数据为实际测量的数据还是预设的报警数值，这个报警数值还可以手动修改，以满足多样化的需求。在最初的设想中，我们想将这个修改预设数值的功能在手机APP和蓝牙模块中实现，在实际实现过程中，受限于我们的知识水平和时间，我们没能及时地编写完整个Android版本的APP，总体代码大约完成了70%，算是本次硬件课设中一个的遗憾。在我们设置的报警机制中，当温湿度没有达到预设的门限时，会启动音乐播放器，通过扬声器播放音乐。一旦温湿度达到或超过预设的报警值时，音乐会立即停止，电风扇被启动降温，同时led灯亮起报警。音乐播放器最开始想通过从U盘中读取相应的音频文件，通过按钮开关实现对音乐播放的暂停、上一曲、下一曲等功能。我们在实际的实现过程中，遇到了无法解码歌曲等问题，所以我们立马寻找其他的代替方法，最后在GitHub中找到类似的音乐播放器功能，通过不同频率、不同占空比的矩形波来模拟Do、Re、Mi、Fa、Sol、La、Si七个音符的高中低音，相当于将FPGA变成一件乐器，将曲子预先写进播放器中的一个变量，未报警的时候就会循环播放音乐，当触发报警机制时，音乐播放器的控制开关会被置为1，输出的矩阵波占空比变为0，也就是不输送任何信号给音频输出端口。

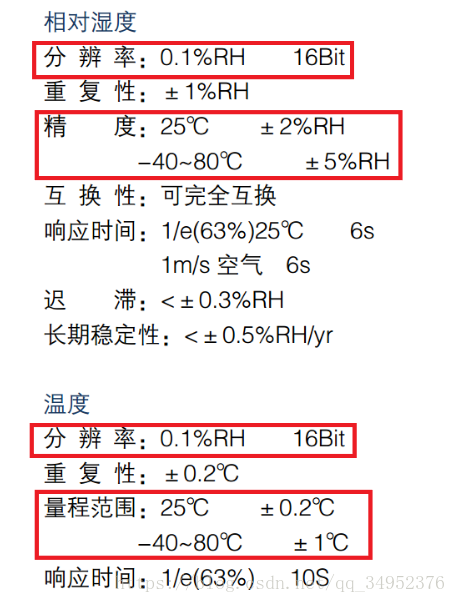
## 3.2结构框图

****

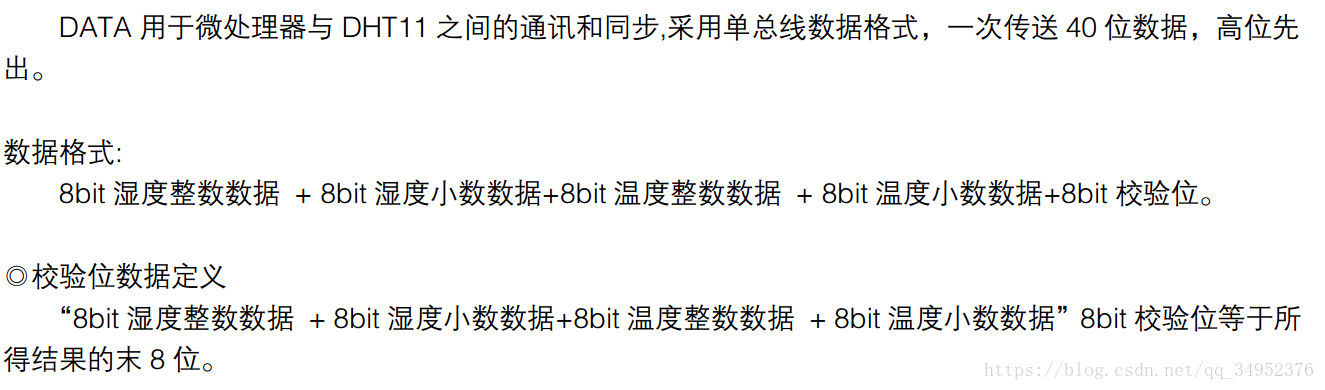
# 课设历程

## 4.1 DHT11介绍

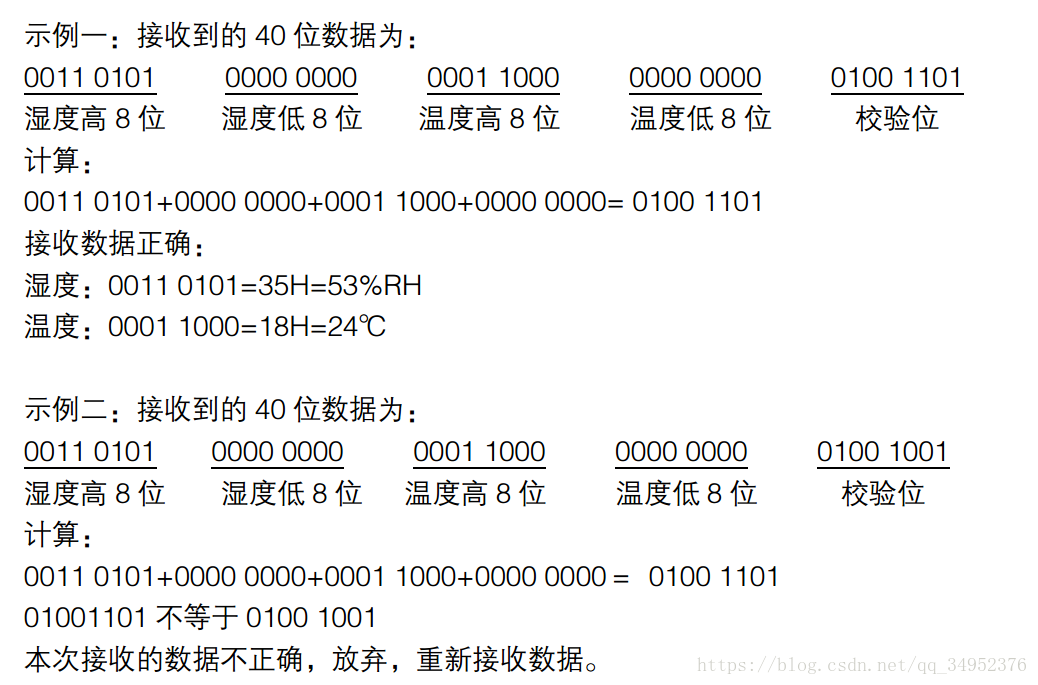
DHT11为国产的温湿度测量模块，提供全中文的参考手册。其中DHT11的分辨率、精度等参数特性如下：



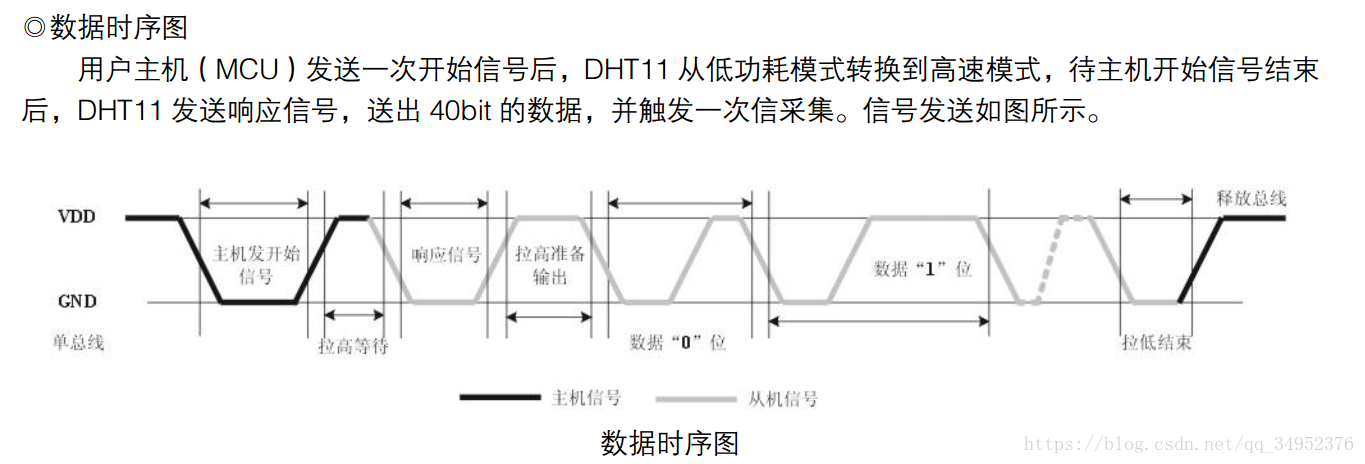
DHT１１仅有三个引脚，分别为VDD、DATA和GND。其中VDD负责供电３.５～５.５VDC，DATA是串行数据总线，单总线。GND接地。其中供电范围为３.５～５.５V，FPGA可以提供相应的电压，满足使用要求。其次，我们发现DHT１１输出的数据格式为：



在数据手册中，还提供了一个校验数据的示例，从中可以更好地理解如何使用DHT１１产生的数据。



知道了产生数据的格式，以及如何校验所得的数据是否正确，剩下的问题就是如何获得数据啦。因为DHT１１是单总线结构，所以需要用户主机发送测量开始信号，DHT１１从低功耗模式转变为高速模式，待主机开始信号结束后，DHT１１发送响应信号，送出４０ｂｉｔ的数据，并且触发一次信采集。整个过程的信号时序图如下：

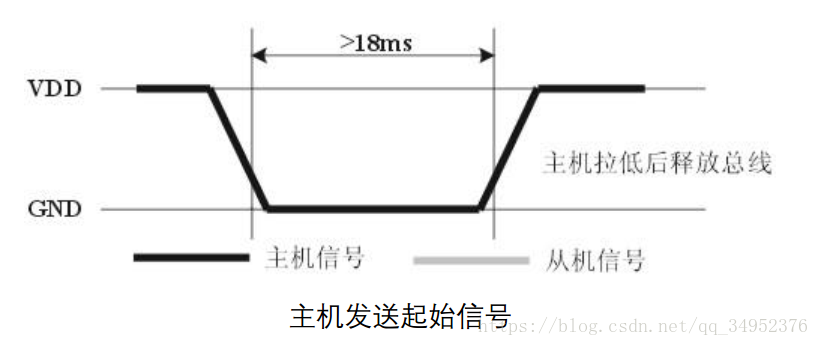


## ４.２DHT１１数据采集过程抽象

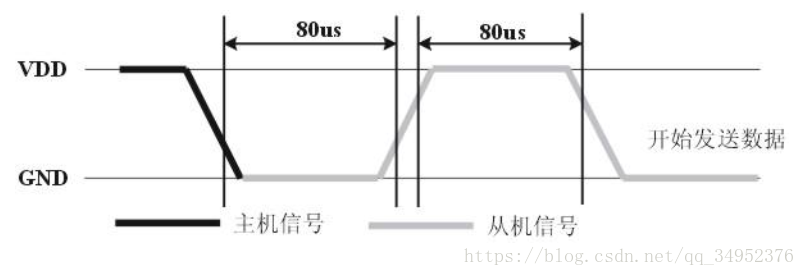
在４.１中已经介绍了如何读取DHT１１产生的数据以及数据的格式，接下来要做的就是对整个过程进行抽象，做出逻辑流程，最后转换为FPGA可以运行的Verilog代码。

步骤一：DHT11 上电后（DHT11 上电后要等待 1S 以越过不稳定状态在此期间不能发送任何指令），测试环境温湿度数据，幵记录数据，同时 DHT11 的 DATA 数据线由上拉电阻拉高一直保持高电平；此时 DHT11 的DATA 引脚处于输入状态，时刻检测外部信号。

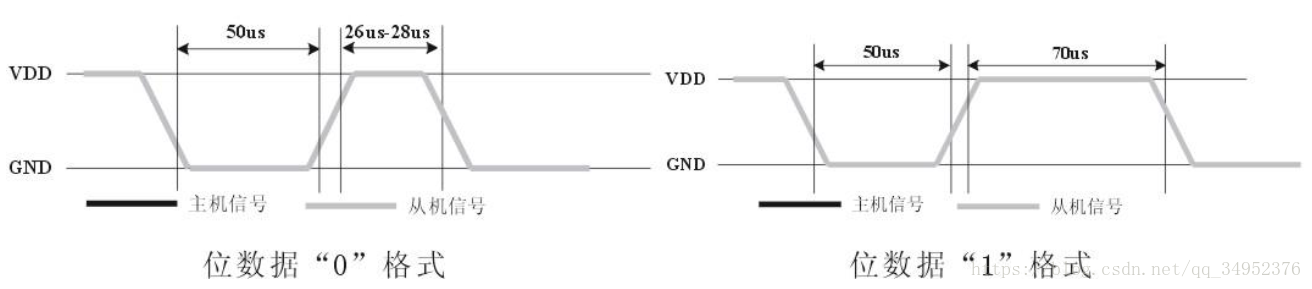
步骤二：微处理器的 I/O 设置为输出同时输出低电平，且低电平保持时间不能小于 18ms，然后微处理器的 I/O设置为输入状态，由于上拉电阻，微处理器的 I/O 即 DHT11 的 DATA 数据线也随之变高，等待 DHT11 作出回答信号，发送信号如图所示：



步骤三：DHT11 的 DATA 引脚检测到外部信号有低电平时，等待外部信号低电平结束，延迟后 DHT11 的 DATA引脚处于输出状态，输出 80 微秒的低电平作为应答信号，紧接着输出 80 微秒的高电平通知外设准备接收数据，微处理器的 I/O 此时处于输入状态，检测到 I/O 有低电平（DHT11 回应信号）后，等待 80 微秒的高电平后的数据接收，发送信号如图所示：



步骤四：由 DHT11 的 DATA 引脚输出 40 位数据，微处理器根据 I/O 电平的变化接收 40 位数据，位数据“0”的格式为： 50 微秒的低电平和 26-28 微秒的高电平，位数据“1”的格式为： 50 微秒的低电平加 70微秒的高电平。位数据“0”、“1”格式信号如图所示：

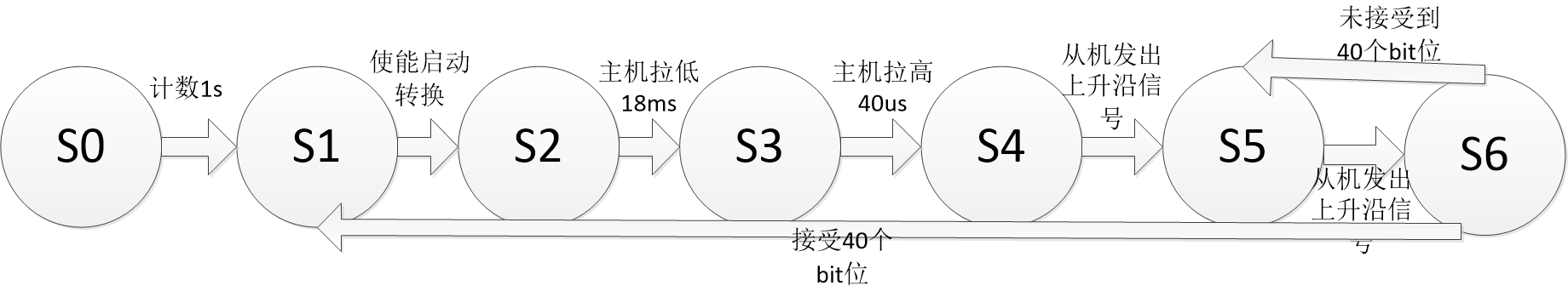


可以把这一段的时序理解为，我们主机先把数据线拉低50us，然后延时等待40us，然后再去读取信号线的电平，如果为低电平，则为位“0”；如果为高电平，则为位“1”

结束信号：DHT11 的 DATA 引脚输出 40 位数据后，继续输出低电平 50 微秒后转为输入状态，由于上拉电阻随之变为高电平。但 DHT11 内部重测环境温湿度数据，幵记录数据，等待外部信号的到来。

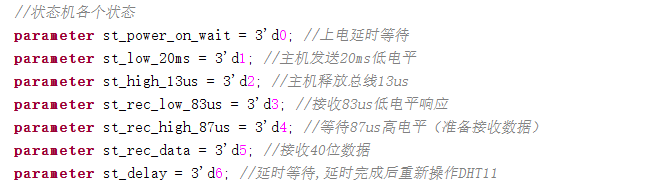
## ４.３温湿度测量代码实现

通过4.2中的分析，我们很容易发现连续的测量过程就是在几个状态中来回切换，自然而然就想到可以用状态机来表示整个过程，以时钟作为触发条件，每次检查当前所处的状态，再结合控制信号可以确定下一个状态。通过状态机所处的不同状态决定相应的操作，如此就可以实现整个DHT１１可控地测量温湿度。构想的状态机如下：

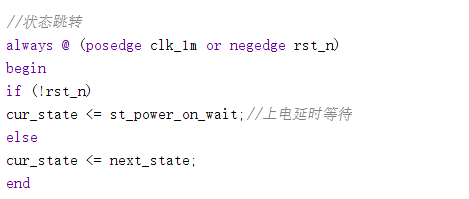


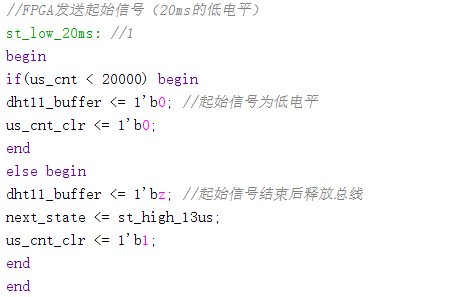
S0：初始状态  
S1：数据接受标志置0  
S2：主机拉低18ms  
S3：主机拉高40us  
S4：从机启动，接收到的40个bit初始化，置0  
S5：从机发送信号  
S6：判断高电平持续时间，如果大于40us，则接受bit位为1并循环

从状态0到状态6总共花费时间粗略在1s以下，因此，对于主机使能信号，1s发送一次，即DHT11采集时间为1s一次就能把40个bit数据全部读出来。在实际代码中，为了调试的方便，将各个状态命名为指示内容的名字，具体代码如下：

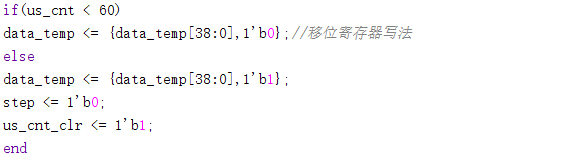


在各个状态下执行相应的操作，并将下一个状态赋值给ｎｅｘｔ＿ｓｔａｔｅ变量，在下一个时钟上升沿到来时再次触发，循环往复，完成对DHT１１的控制。完整代码参考七中的代码，下面给出状态跳转和一个状态下操作的代码。

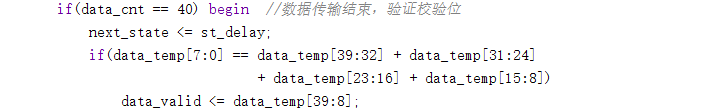




通过输出信号在高电平维持的时间长短来判断接收数据为1还是为0，具体代码如下：



通过DHT１１送出的最后８位校验码来校验所得数据是否准确，如果不满足校验条件，不传输显示，具体代码如下：



## 4.4学习过程回顾

### 4.4.1资料整理

最开始老师推荐使用的是STM32魔女开发板，因为之前没使用过，所以感觉上手难度大于直接在微机原理中使用过的NEXYS4FPGA开发板。在经过老师的同意后，选用了FPGA作为开发的硬件，当时因为贪图简单，没有仔细思考过FPGA的外界模块仅能通过5个PMOD接口连接，而且没有时序逻辑，时钟频率也只有100M的频率，对外界模块的限制比较大。

在最开始的一周准备工作的时候，我逐渐发现，关于智能家居部分的资料虽然不少，但大都基于单片机或者STM32，使用FPGA来实现智能家居模块对我们来说是个不小的挑战。在最后找不到实例的情况下，我决定先使用官方开发文档，对FPGANEXYS4开发板进行熟悉，复习之前掌握的Verilog语言的基本语法规则。然后我打算在STM32做外界模块的扩展的基础上，学习如何操纵外界模块，将整个过程抽象出来，接着在FPGA上实现。因此我主要搜集到了Xlinx官方开发文档和CSDN有关博客的介绍作为参考资料。

### 4.4.2曲折的学习过程

因为借鉴的是STM32的开发实例，所以代码部分完全不能参考，一开始走了许多弯路，DHT11模块最开始一直无法正常工作，时钟分频也得不到想要的频率，项目进展陷入困境。之前在做微机原理实验的时候，因为频率要求比较高，所以使用IP核就可以满足模块的需求，但碰到需要时钟频率要求较低的DHT１１模块，时钟的周期需要达到１ｍｓ，但１００M的板载时钟通过IPｃｏｒｅ的分频最多可以降低到１MＨｚ，无法满足模块运行的需求。后来在网上查阅资料后发现，可以通过偶分频和奇分频得到想要的时钟频率，这让我喜出望外，通过仿真和实际验证，发现可以得到想要的时钟频率。（验证部分采用的LED闪烁的方法，发现可以得到符合要求的时钟频率），在将DHT１１读取数据的过程抽象之后，我开始采用状态机的方式来解决这个问题，最后可以将正确的温湿度数据读取后发送到数码管显示出来。

### ４.４.３回顾分析

整个硬件课设的过程并不容易，起初我们贪图使用板子的简易选择了FPGA，但后续资料的匮乏一度让我陷入困境，无法快速实现相关的功能。在实现完温湿度的测量后，我开始着手于数码管的显示，这里沿用了我之前编写过的数码管显示代码，但后续又碰到问题，数码管智能显示BCD码，可DHT１１送出的数据为８位二进制，如何将其分为个位和十位成了新的问题。查阅资料之后，我发现可以使用一个时钟频率远高于显示频率的时钟来作一个加法器，每次将得到的数据和原来采集得到的数据进行比较，满足相等条件停止增加。由此可以实现二进制转换为BCD码，在实现报警模块的时候需要通过一个三目运算符来判断是否显示预设温湿度，通过温湿度比较判断是否触发报警机制。报警机制最开始的时候打算使用电风扇和LED来实现，但电风扇的电流过大，导致每次触发报警的时候，电风扇一运转就会触发板子的保护机制，随意后面又选用了音乐播放模块来完善报警机制，未触发的时候播放音乐，出发之后停止播放，通过预先编写入板子的曲子实现歌曲的播放。

# 综合调试结果

## ５.１　DHT１１温湿度测量模块调试

在对温湿度测量模块进行调试时，借助了数码管的显示。因为DHT11输出的40位数据中，有３２位为数据位，分别为温湿度的８为整数和８位小数，首先要将８位数据转变为数码管可以显示的BCD码，后续的调试过程才能顺利进行。也就是说最开始，先调试数码管的显示模块。

如何将８位的二进制转化为个位和十位的BCD码就成了首要问题，我们后来想到可以用一个比数码管刷新频率高足够多的频率来进行一个加法，初始的数据有00开始增加，每次时钟上升沿到来时加一，并于8位二进制数进行比较，如果个位数字大于9就让十位进一，当两个数字相等时加法结束。理论上应该设置百位等高位，来应对任何情况，但结合4.1中DHT11的参数介绍，发现测量范围并没有那么宽，所以十位已经足够使用。将转换后的BCD码输送给数码管显示，然后与实际的室温相比较，可以得知控制DHT１１的相关代码是否正常工作。

## ５.２音乐播放模块调试

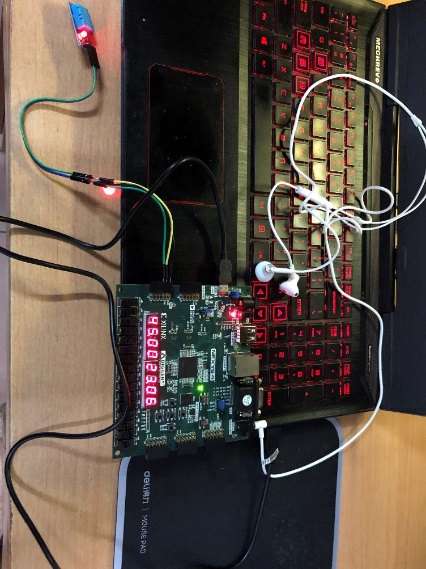
另外新建了音乐播放的工程，避免调试过程对已经实现功能的代码的影响。在将音乐播放模块相关代码完成后，我们开始对音乐播放模块的调试，最开始一直没有声音，经过仔细排查，发现FPGA上音频输出端口与耳机接口需要在一个合适的连接位置才能正常输出，一开始耳机插入端口过深，没有任何信号输出。在调整接口的连接后，我们发现使能开关可以正常工作，可以控制音乐的播放和暂停。但音乐始终发出同一个声音，与我们预想的乐曲有明显的区别，排查完物理上的问题，我们将目光重新放回音乐播放器的代码上，经过逐行检查，发现最开始调试声音的时候将输出的矩形波设为一个固定的值，后续没有及时修改回正确的版本，修改完成后，音乐播放器模块可以正常工作。

## ５.３预设参数修改模块调试

我们设想可以通过FPGA上的开关或者按钮来对预设的报警数值进行修改，所以修改的触发条件设置为按钮的下降沿，即按一次开关进行一次修改。分模块缺少了数据来源，所以数据就只能通过FPGA上的开关来输入，通过LED灯　来显示转换后的结果。最开始将增加和减少放在同一个函数中，Verilog不允许在同一个module中一个变量存在ｍｕｌｔｉｐｌｅ　ｄｒｉｖｅ，所以将增减分为两个函数，分别实现。在实际调试的过程中发现使用按钮可以较为完美地实现增减过程，但在整体调试时发现，按钮的引脚分配和时钟信号的引脚分配产生冲突，所以换成了开关来模拟按钮，同时出现开关接触导致出现多次下降沿。

## ５.４整机系统调试

整个系统调试的过程较为顺利，最后的实物图片如下：



# 课设总结

## ６.１课设分工

|  |  |
| --- | --- |
| 小组成员 | 分工 |
| 张许（组长） | DHT11温湿度测量相关代码 |
| 杨深镇 | 音乐播放器模块相关代码 |
| 周馨瑞 | 数码管、预设温湿度修改相关代码 |

## 6.2部分遇到的问题及解决方案

(1)调试DHT11时，8位整数如何拆分为十位和个位？

这个问题受到CSDN中一个人思想的启发，只要在显示数据（需要数据）之前，将数据修改为正确的形式就可以了，所以通过一个加法来实现十位和个位的分离。

（2）音频文件需要解码，FPGA需要借助外接模块实现。

我们后来通过将FPGA变成一个乐器，同样实现了音乐播放的功能。

（3）预设数值修改，一直没有反应。

排查后发现是Wire和Reg类型的原因，然后将增加和减小分为两个module，将触发条件变为数据的改变，减小运算资源的消耗。

## 6.3未来展望

目前我们完成的系统，与我们最开始设想的系统还是有一定差距的，例如通过YourCeeM38蓝牙模块，实现手机APP控制整个系统，达到远程控制开关、修改数值的功能。其次可以配合其他的外接模块，完成温湿度的播报。

## 6.4心得体会

因为疫情的影响，硬件课设晚了一个学期才开始。因为绝大多数相关的资料都是基于STM32的，所以整个工程开展起来的难度还是不小的。在整个过程中，我们遇到了很多困难和问题，其中有些是很低级的，有些则是原理上、编程语言等层面的。等到最后成品运行的那一刻，我就感觉到满满的成就感，这次是迄今为止最为完整的一次项目，从前期的选题、资料搜集到最后的整机调试、报告撰写，我都感觉十分充实。

感谢潘老师一路走来的陪伴，更感谢他所给予的帮助和督促。

# 七、附录代码

## ７.１顶层模块

module top(

input clk\_100,

input reset,//time\_control

input reset\_t,//temperature\_control

input reset\_s,

input con\_d,

input con\_c,//控制预设温度

input pb,

input db,

inout dht11,

output music,

output [3:0]led,

output [7:0] an, // 数码管使能

output [7:0] seg// 数码管输出

);

wire clk\_8;

wire clk\_1;

wire clk\_50;

reg [2:0] count;

wire [31:0] data;

wire [3:0] data1,data2,data3,data4,data5,data6,data7,data8,d5,d6,t5,t6;

wire[3:0] data5\_p = 4'b0010;

wire[3:0] data6\_p = 4'b0101;

clk\_wiz\_1(.clk\_in1(clk\_100),.reset(reset),.clk\_out1(clk\_50));

//time\_adv\_even(.clk(clk\_8),.rst(reset),.clk\_out(clk\_1));//得到1M时钟dht11\_drive(.sys\_clk(clk\_50),.rst\_n(reset\_t),.dht11(dht11),.data\_valid(data));

bin\_dec(.clk(clk\_100),.bin(data[31:24]),.rst\_n(reset\_t),.one(data2),.ten(data1));//二进制转换为十进制BCD

bin\_dec(.clk(clk\_100),.bin(data[15:8]),.rst\_n(reset\_t),.one(data8),.ten(data7));

Wchange(data5\_p,data6\_p,con\_c,pb,t5,t6);

Wchange2(t5,t6,con\_c,db,d5,d6);

Wdisplay(d5,d6,data7,data8,con\_d,data5,data6); //Wdisplay(data5\_p,data6\_p,data7,data8,con\_d,data5,data6);

//alert(clk\_100,data5\_p,data6\_p,data7,data8,music,led);

alert(clk\_100,d5,d6,data7,data8,music,led);

seg(.reset(reset\_s),.clock(clk\_100),.seg(seg),.an(an),.score\_a(data1),.score\_b(data2),.score\_c(data[23:20]),.score\_d(data[19:16]),.score\_e(data5),.score\_f(data6),.score\_g(data[7:4]),.score\_h(data[3:0]));

endmodule

## ７.２DHT１１温湿度测量模块

module dht11\_drive(

input sys\_clk,

input rst\_n,

inout dht11,//dht11单总线

output reg [31:0] data\_valid //有效数据

);

//parameter define

parameter POWER\_ON\_NUM = 1000\_000; //上电延时等待时间,单位us

//状态机各个状态

parameter st\_power\_on\_wait = 3'd0; //上电延时等待

parameter st\_low\_20ms = 3'd1; //主机发送20ms低电平

parameter st\_high\_13us = 3'd2; //主机释放总线13us

parameter st\_rec\_low\_83us = 3'd3; //接收83us低电平响应

parameter st\_rec\_high\_87us = 3'd4; //等待87us高电平（准备接收数据）

parameter st\_rec\_data = 3'd5; //接收40位数据

parameter st\_delay = 3'd6; //延时等待,延时完成后重新操作DHT11

//reg define

reg [2:0] cur\_state ; //当前状态

reg [2:0] next\_state ; //下一个状态

reg [4:0] clk\_cnt ; //分频计数器

reg clk\_1m ; //1Mhz时钟

reg [20:0] us\_cnt ; //1微秒计数器

reg us\_cnt\_clr ; //1微秒计数器清零信号

reg [39:0] data\_temp ; //缓存接收到的数据

reg step ; //数据采集状态

reg [5:0] data\_cnt ; //接收数据用计数器

reg dht11\_buffer; //DHT11输出信号

reg dht11\_d0 ; //DHT11输入信号寄存器0

reg dht11\_d1 ; //DHT11输入信号寄存器1

//wire define

wire dht11\_pos ; //DHT11上升沿

wire dht11\_neg ; //DHT11下降沿

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\* main code

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

assign dht11 = dht11\_buffer;

assign dht11\_pos = ~dht11\_d1 & dht11\_d0; //采集上升沿

assign dht11\_neg = dht11\_d1 & ~dht11\_d0; //采集下降沿

//得到1Mhz分频时钟

//时钟为50M通过偶分频得到1M时钟即50分频

//用计数器从0技术到50/2-1 偶分频N/2-1s

//该项目使用1M分频是因为DHT11时序主要以us为单位

always @ (posedge sys\_clk or negedge rst\_n)

begin

if (!rst\_n)

begin

clk\_cnt <= 5'd0;

clk\_1m <= 1'b0;

end

else if (clk\_cnt < 5'd24)

clk\_cnt <= clk\_cnt + 1'b1;

else begin

clk\_cnt <= 5'd0;

clk\_1m <= ~ clk\_1m;

end

end

//对DHT11输入信号连续寄存两次，用于边沿检测

always @ (posedge clk\_1m or negedge rst\_n) begin

if (!rst\_n) begin

dht11\_d0 <= 1'b1;

dht11\_d1 <= 1'b1;

end

else begin

dht11\_d0 <= dht11;

dht11\_d1 <= dht11\_d0;

end

end

//1微秒计数器

always @ (posedge clk\_1m or negedge rst\_n) begin

if (!rst\_n)

us\_cnt <= 21'd0;

else if (us\_cnt\_clr)

us\_cnt <= 21'd0;

else

us\_cnt <= us\_cnt + 1'b1;

end

//状态跳转

always @ (posedge clk\_1m or negedge rst\_n)

begin

if (!rst\_n)

cur\_state <= st\_power\_on\_wait;//上电延时等待

else

cur\_state <= next\_state;

end

//状态机读取DHT11数据

always @ (posedge clk\_1m or negedge rst\_n)

begin

if(!rst\_n)

begin

next\_state <= st\_power\_on\_wait;

data\_temp <= 40'd0;

step <= 1'b0;

us\_cnt\_clr <= 1'b0;

data\_cnt <= 6'd0;

dht11\_buffer <= 1'bz;

end

else

begin

case (cur\_state)

//上电后延时1秒等待DHT11稳定

st\_power\_on\_wait:

begin

if(us\_cnt < POWER\_ON\_NUM) //没有达到上电延迟时间1s

begin

dht11\_buffer <= 1'bz; //空闲状态释放总线

us\_cnt\_clr <= 1'b0;

end

else

begin //进入下一状态，清零us计数器

next\_state <= st\_low\_20ms;

us\_cnt\_clr <= 1'b1;

end

end

//FPGA发送起始信号（20ms的低电平）

st\_low\_20ms: //1

begin

if(us\_cnt < 20000) begin

dht11\_buffer <= 1'b0; //起始信号为低电平

us\_cnt\_clr <= 1'b0;

end

else begin

dht11\_buffer <= 1'bz; //起始信号结束后释放总线

next\_state <= st\_high\_13us;

us\_cnt\_clr <= 1'b1;

end

end

//等待DHT11的响应信号（等待10~20us）

st\_high\_13us://2

begin

if (us\_cnt < 20) begin

us\_cnt\_clr <= 1'b0;

if(dht11\_neg)

begin //检测到DHT11响应信号

next\_state <= st\_rec\_low\_83us;

us\_cnt\_clr <= 1'b1;

end

end

else //超过20us未响应，重新进入延时

next\_state <= st\_delay;

end

//等待DHT11的83us低电平响应信号结束

st\_rec\_low\_83us: //3

begin

if(dht11\_pos)

next\_state <= st\_rec\_high\_87us;

end

//DHT11拉高87us通知FPGA准备接收数据

st\_rec\_high\_87us: begin

if(dht11\_neg) begin //准备时间结束

next\_state <= st\_rec\_data;

us\_cnt\_clr <= 1'b1;

end

else begin //高电平准备接收数据

data\_cnt <= 6'd0;

data\_temp <= 40'd0;

step <= 1'b0;

end

end

//连续接收40位数据

st\_rec\_data: begin

case(step)

0: begin //接收数据低电平

if(dht11\_pos) begin

step <= 1'b1;

us\_cnt\_clr <= 1'b1;

end

else //等待数据低电平结束

us\_cnt\_clr <= 1'b0;

end

1: begin //接收数据高电平

if(dht11\_neg) begin

data\_cnt <= data\_cnt + 1'b1;

//判断接收数据为0/1

if(us\_cnt < 60)

data\_temp <= {data\_temp[38:0],1'b0};//移位寄存器写法

else

data\_temp <= {data\_temp[38:0],1'b1};

step <= 1'b0;

us\_cnt\_clr <= 1'b1;

end

else //等待数据高电平结束

us\_cnt\_clr <= 1'b0;

end

endcase

if(data\_cnt == 40) begin //数据传输结束，验证校验位

next\_state <= st\_delay;

if(data\_temp[7:0] == data\_temp[39:32] + data\_temp[31:24]

+ data\_temp[23:16] + data\_temp[15:8])

data\_valid <= data\_temp[39:8];

end

end

//完成一次数据采集后延时2s

st\_delay:begin

if(us\_cnt < 2000\_000)

us\_cnt\_clr <= 1'b0;

else begin //延时结束后重新发送起始信号

next\_state <= st\_low\_20ms;

us\_cnt\_clr <= 1'b1;

end

end

default : ;

endcase

end

end

endmodule

## ７.３报警模块

module alert(

input clk,

input [3:0]d1,

input [3:0]d2,

input [3:0]d3,

input [3:0]d4,

output out,

output [3:0]led

);

wire c1;

reg c2;

always @(d1 or d2 or d3 or d4)

begin

if(d3 > d1)

c2 = 1'b1;

else if((d4 >= d2)&&(d1 == d3))

c2 = 1'b1;

else

c2 = 1'b0;

end

assign c1 = c2;

music(clk,c1,out,led);

endmodule

## ７.４音乐播放模块

module music(input clk,

input rst,

output out,

output [3:0]led

);

reg [4:0]temp;

reg [3:0]ledtemp = 4'b0000;

wire [119:0]music;

reg [9:0]count = 10'd0;

wire clk\_play;

Divider #(50\_000\_000) s0(.I\_CLK(clk),.Rst(rst),.O\_CLK(clk\_play));//一秒2个音符

assign music[4:0] = 5'b01100;

assign music[9:5] = 5'b01010;

assign music[14:10] = 5'b01100;

assign music[19:15] = 5'b01010;

assign music[24:20] = 5'b01100;

assign music[29:25] = 5'b01010;

assign music[34:30] = 5'b01000;

assign music[39:35] = 5'b01000;

assign music[44:40] = 5'b01001;

assign music[49:45] = 5'b01011;

assign music[54:50] = 5'b01010;

assign music[59:55] = 5'b01001;

assign music[64:60] = 5'b01100;

assign music[69:65] = 5'b01100;

assign music[74:70] = 5'b01100;

assign music[79:75] = 5'b01100;

assign music[84:80] = 5'b01100;

assign music[89:85] = 5'b01010;

assign music[94:90] = 5'b01100;

assign music[99:95] = 5'b01010;

assign music[104:100] = 5'b01100;

assign music[109:105] = 5'b01010;

assign music[114:110] = 5'b01000;

assign music[119:115] = 5'b01000;

always @(posedge clk\_play)

begin

temp[4:0] <= music[count +4 -:5];

end

always @(posedge clk\_play)

begin

count <= count + 5;

if(count == 120)

count <= 0;

end

Play(.clk(clk),.rst(rst),.out(out),.sw(temp));

assign led[0] = rst;

assign led[3] = rst;

assign led[2] = rst;

assign led[1] = rst;

endmodule

module Divider #(parameter times=20)(I\_CLK,Rst,O\_CLK);

input I\_CLK,Rst;

output O\_CLK;

reg temp=1'b0;

localparam tmp\_times=times/2;

assign O\_CLK=temp;

integer cnt=0;

always @(posedge I\_CLK)

begin

if(Rst==1'b1)

begin

cnt<=0;

temp<=1'b0;

end

else

begin

if(cnt==tmp\_times-1)

begin

temp<=~temp;

cnt<=0;

end

else

cnt<=cnt+1;

end

end

endmodule

module Play(

input clk,

input [4:0]sw,

input rst,

output out

);

localparam l\_do=339\_732,

l\_ri=302\_668,

l\_mi=269\_560,

l\_fa=258\_666,

l\_so=253\_334,

l\_la=201\_866,

l\_si=179\_998,

m\_do=169\_866,

m\_ri=156\_534,

m\_mi=134\_666,

m\_fa=125\_600,

m\_so=113\_334,

m\_la=101\_006,

m\_si=88\_000,

h\_do=85\_066,

h\_ri=73\_866,

h\_mi=66\_134,

h\_fa=62\_134,

h\_so=55\_466,

h\_la=49\_333,

h\_si=44\_000,

stop=266;

reg [4:0]cnt;

always@(posedge clk)

case(sw)

5'd0:cnt<=5'd0;

5'd1:cnt<=5'd1;

5'd2:cnt<=5'd2;

5'd3:cnt<=5'd3;

5'd4:cnt<=5'd4;

5'd5:cnt<=5'd5;

5'd6:cnt<=5'd6;

5'd7:cnt<=5'd7;

5'd8:cnt<=5'd8;

5'd9:cnt<=5'd9;

5'd10:cnt<=5'd10;

5'd11:cnt<=5'd11;

5'd12:cnt<=5'd12;

5'd13:cnt<=5'd13;

5'd14:cnt<=5'd14;

5'd15:cnt<=5'd15;

5'd16:cnt<=5'd16;

5'd17:cnt<=5'd17;

5'd18:cnt<=5'd18;

5'd19:cnt<=5'd19;

5'd20:cnt<=5'd20;

5'd21:cnt<=5'd21;

default:cnt<=5'd0;

endcase

assign out=mclk[cnt];

wire mclk[21:0];

Divider #(stop) s0(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[0]));

Divider #(l\_do) l1(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[1]));

Divider #(l\_ri) l2(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[2]));

Divider #(l\_mi) l3(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[3]));

Divider #(l\_fa) l4(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[4]));

Divider #(l\_so) l5(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[5]));

Divider #(l\_la) l6(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[6]));

Divider #(l\_si) l7(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[7]));

Divider #(m\_do) m1(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[8]));

Divider #(m\_ri) m2(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[9]));

Divider #(m\_mi) m3(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[10]));

Divider #(m\_fa) m4(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[11]));

Divider #(m\_so) m5(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[12]));

Divider #(m\_la) m6(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[13]));

Divider #(m\_si) m7(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[14]));

Divider #(h\_do) h1(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[15]));

Divider #(h\_ri) h2(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[16]));

Divider #(h\_mi) h3(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[17]));

Divider #(h\_fa) h4(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[18]));

Divider #(h\_so) h5(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[19]));

Divider #(h\_la) h6(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[20]));

Divider #(h\_si) h7(.I\_CLK(clk),.Rst(rst),.O\_CLK(mclk[21]));

Endmodule

module Divider #(parameter times=20)(I\_CLK,Rst,O\_CLK);

input I\_CLK,Rst;

output O\_CLK;

reg temp=1'b0;

localparam tmp\_times=times/2;

assign O\_CLK=temp;

integer cnt=0;

always @(posedge I\_CLK)

begin

if(Rst==1'b1)

begin

cnt<=0;

temp<=1'b0;

end

else

begin

if(cnt==tmp\_times-1)

begin

temp<=~temp;

cnt<=0;

end

else

cnt<=cnt+1;

end

end

endmodule

## ７.５增减模块和选择显示模块

module Wchange(

input [3:0] d1,

input [3:0] d2,

input con,

input pbutton,

output [3:0]d3,

output [3:0]d4

);

reg [3:0] t1 = 4'b0000;

reg [3:0] t2 = 4'b0000;

always @(negedge pbutton)

begin

if(con == 1'b1)

t2 <= t2 + 4'b0001;

else

t1 <= t1 + 4'b0001;

end

module Wchange2(

input [3:0] d1,

input [3:0] d2,

input con,

input dbutton,

output [3:0]d3,

output [3:0]d4

);

reg [3:0] t1 = 4'b0000;

reg [3:0] t2 = 4'b0000;

always @(negedge dbutton)

begin

if(con == 1'b1)

t2 <= t2 - 4'b0001;

else

t1 <= t1 - 4'b0001;

end

assign d3 = d1 + t1;

assign d4 = d2 + t2;

endmodule

module Wdisplay(

input [3:0]d1,

input [3:0]d2,

input [3:0]d3,

input [3:0]d4,

input con,

output [3:0]d5,

output [3:0]d6

);

assign d5 = (con) ? d1:d3;

assign d6 = (con) ? d2:d4;

endmodule

## ７.６数码管显示模块

module seg(

// 变量说明见top模块

input reset,

input clock,

input [3:0] score\_a,score\_b,score\_c,score\_d,score\_e,score\_f,score\_g,score\_h,

output reg [7:0] seg,

output reg [7:0] an

);

wire [7:0] a\_seg,b\_seg,c\_seg,d\_seg,e\_seg,f\_seg,g\_seg,h\_seg;

bcdto8segment\_dataflow (.x(score\_a),.seg(a\_seg));

bcdto8segment\_dataflow (.x(score\_b),.seg(b\_seg));

bcdto8segment\_dataflow (.x(score\_c),.seg(c\_seg));

bcdto8segment\_dataflow (.x(score\_d),.seg(d\_seg));

bcdto8segment\_dataflow (.x(score\_e),.seg(e\_seg));

bcdto8segment\_dataflow (.x(score\_f),.seg(f\_seg));

bcdto8segment\_dataflow (.x(score\_g),.seg(g\_seg));

bcdto8segment\_dataflow (.x(score\_h),.seg(h\_seg));

reg [37:0] count;

initial

begin

count<=0;

end

always @(posedge clock or posedge reset)

begin

if (reset==1)

begin

an<=8'b00000000;

seg<=8'b10000000;

count<=0;

end

else

begin

// count每数100000切换数码管，相当于每次切换的时间为100k/100M=1/1000s=1ms

if (count==800000)

begin

an<=8'b01111111;

seg<=a\_seg;

count<=0;

end

else if (count==700000)

begin

an<=8'b10111111;

seg<=b\_seg&8'b01111111;//添加小数点

count<=count+1;

end

else if (count==600000)

begin

an<=8'b11011111;

seg<=c\_seg;

count<=count+1;

end

else if (count==500000)

begin

an<=8'b11101111;

seg<=d\_seg;

count<=count+1;

end

else if (count==400000)

begin

an<=8'b11110111;

seg<=e\_seg;

count<=count+1;

end

else if (count==300000)

begin

an<=8'b11111011;

seg<=f\_seg&8'b01111111;

count<=count+1;

end

else if (count==200000)

begin

an<=8'b11111101;

seg<=g\_seg;

count<=count+1;

end

else if (count==100000)

begin

an<=8'b11111110;

seg<=h\_seg;

count<=count+1;

end

else count<=count+1;

end

end

endmodule

module bcdto8segment\_dataflow(

input [3:0] x,

output reg [7:0] seg

);

always @(x)

begin

case(x)

4'b0000 : seg <= 8'b11000000;

4'b0001 : seg <= 8'b11111001;

4'b0010 : seg <= 8'b10100100;

4'b0011 : seg <= 8'b10110000;

4'b0100 : seg <= 8'b10011001;

4'b0101 : seg <= 8'b10010010;

4'b0110 : seg <= 8'b10000010;

4'b0111 : seg <= 8'b11111000;

4'b1000 : seg <= 8'b10000000;

4'b1001 : seg <= 8'b10010000;

default : seg <= 8'b1xxxxxxx;

endcase

end