**汇编与接口课程设计**

**实验报告**

**简单接口控制模块设计——打砖块**

**小组成员**

**学号 姓名**

**学号 姓名**

**学号 姓名**

**班级：071XXXXX**

**简单接口控制模块设计——打砖块**

1. **设计目的**
2. 知悉和理解现代接口技术
3. 知悉现代数字系统设计方法和基本流程，并学习一门硬件描述语言
4. 掌握常用接口硬件的功能、特点及使用方法
5. 能够读懂简单的接口硬件原理图，根据要求设计接口控制模块
6. **设计内容**

设计一个 VGA 显示控制器，支持 640×480×60Hz。

VGA显示打方块游戏，通过开发板上的按键来控制挡板的左右移动，小球碰到挡板能够反弹，碰到砖块则砖块消失，小球反弹。如果挡板没有接到小球，则游戏结束，需要手动重置重新进行游戏。

1. **设备器材**

操作系统：Windows 10

开发环境：Vivado

开发板：Xilinx Artix-7系列开发板

1. **成员与分工**

|  |  |  |  |
| --- | --- | --- | --- |
| 成员 | 分工 | 完成情况 |  |
| A（组长） | 编写各模块代码实现，撰写报告 | 完成 |  |
| B | 实现按键消抖，撰写报告 | 完成 |  |
| C | 将图片转换为coe文件，撰写报告 | 完成 |  |

组长联系方式：134XXXXX

1. **设计原理及内容**

VGA (Video Graphics Array，视频图形阵列)，是IBM于1987年出的一个使用模拟信号的电脑显示标准。VGA仍是目前最多制造商所共同支持的一个标准，个人电脑在加载自己的独特驱动程序之前，都必须支持VGA的标准。VGA 最早支持在640X480的较高分辨率下同时显示16种色彩或256种灰度，或者在 320X240分辨率下可以同时显示256种颜色。VGA接口共有15针，分成三排，每排5个孔，是目前显卡、显示器等设备上应用最广泛的接口类型。在FPGA中，可以通过构建VGA时序电路模块来生成对应的VGA时序信号，同时将生成的或预先储存的图像数据进行输出显示。VGA接口就是显卡上输出模拟信号的接口，VGA接口，也叫D-Sub接口，如下图所示。

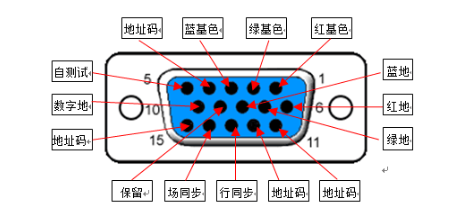


图5-1

在本次实验中，通过VGA接口将存储在ROM中的图片进行输出，并且可以通过不断改变每帧图像中图片的显示位置使VGA输出的图像为动态图像，即达到小球不断运动的效果以及实现挡板的运动。同时，通过控制一个变量的每一位来实现砖块的显示与否，实验中使用 FPGA 内部时钟管理模块产生系统所需时钟；使用ROM IP核来存储事先准备好的图片数据；最后通过 VGA 时序模块产生时序并连同数据一同输出至VGA显示器上。

VGA 时序控制

通常 VGA 显示器扫描方式分为逐行扫描和隔行扫描。

逐行扫描是扫描从屏幕左上角一点开始，从左向右逐点扫描，每扫描完一行，电子束回到屏幕的左边下一行的起始位置。在这期间，需要对电子束进行消隐。每行结束时，用行同步信号进行同步。当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描点回到屏幕左上方，同时进行场消隐并开始下一帧。隔行扫描是指电子束扫描时每隔一行扫一线，完成一屏后再返回来扫描剩下的线。完成一行扫描的时间称为水平扫描时间，其倒数称为行频率。完成一帧(整屏)扫描的时间称为垂直扫描时间，其倒数称为场频率，即刷新一屏的频率，常见的有 60Hz、75Hz 等等。标准的VGA显示的场频为60Hz，行频为31.5KHz。如图 5-2 所示，一个完整的行扫描周期由a、b、c、d四部分组成。即：

1. 行扫描地址的复位；
2. 扫描地址转移后的稳定等待/准备期；
3. 行地址扫描/显示期，此时数据有效；
4. 扫描地址转移的准备；

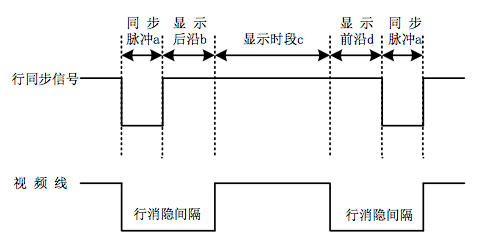


图5-2 VGA行扫描

一个场扫描周期由n个行扫描周期组成，并且一次场扫描时序自成体系，有完整的场消隐前肩、场同步时期、场消隐后肩、场显示时期。时序如图 5-3中场时序所示，与行扫描类似有以下几个部分：

1. 场扫描地址的复位；
2. 扫描地址转移后的稳定等待/准备期；
3. 场地址扫描/显示期，此时数据有效；
4. 扫描地址转移的准备；

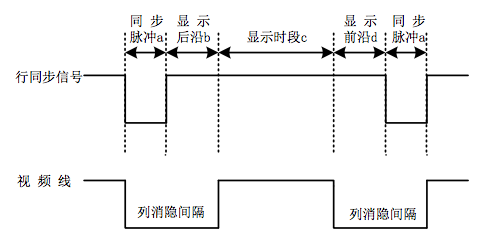


图5-3 VGA场扫描

在实验中使用的是 640×480 分辨率

1. **设计步骤**

## 确定设计目标

在课程任务书中老师给出了几个可选的项目，但是经过小组讨论之后，我们决定自行设计一个小游戏。最开始讨论的结果是做一个贪吃蛇或者俄罗斯方块游戏，但是考虑到已经有人做过此类游戏，最后没有采用。

在上网查阅了一番资料并且结合我们小组成员的经历之后，我们最终决定做一个小时候常在游戏机或者手机上玩的一个“打砖块”游戏。

## 讨论设计思路

在决定做一个“打砖块”游戏之后，我们小组进行了一个设计思路的讨论，比如如何实现砖块的显示，如何实现小球碰撞反弹的效果以及如何控制挡板的移动等。最终我们将砖块的显示信息存入一个140位的变量中，其中每一位代表一块砖，一共140块砖，1代表显示，0代表不显示，在砖块显示模块，根据当前扫描到的区域对应的砖块来确定是否显示。

我们给小球、砖块、挡板都设定了坐标，所有的显示我们都根据当前扫描到的点和坐标做比对来确定当前点在哪个区域内，如果在砖块区域，就去查表是否显示砖块；如果是在小球或挡板区域内就显示小球或挡板。

至于小球的移动，我们决定再加一个运动速度的变量，每一定数量个时钟周期小球的当前坐标和运动速度做运算即可得到小球的移动，挡板的控制比较好做，每次检测到按键，就将挡板的坐标进行移动即可。

## 确定整个游戏的布局

因为我们通过坐标和当前扫描点的比对来控制显示，所以需要事先画出屏幕的框架图：

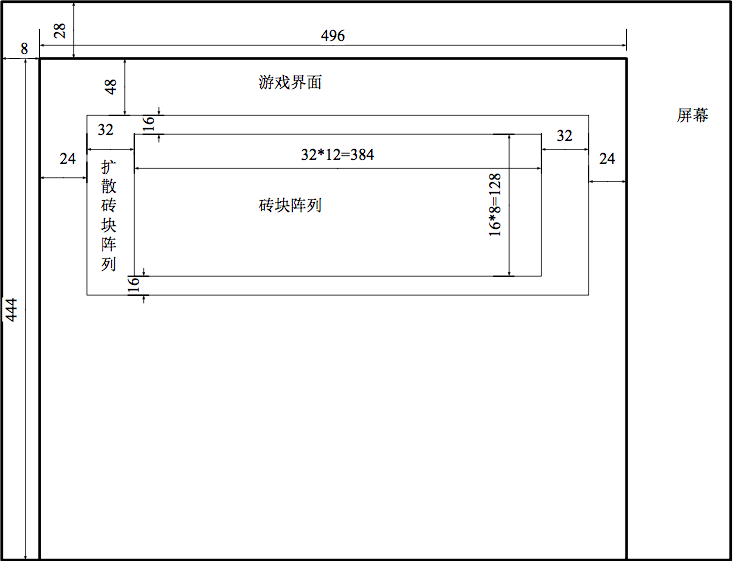


图6-1 屏幕框架图

框架图中，左上角为屏幕的左上角顶点，也是整个坐标系的原点，游戏界面并不是整个屏幕，而是屏幕中靠左下的一大块区域，右边区域空了出来，是为了显示一些游戏信息的，但是因为时间关系所以没有用上。

## 代码实现

整个项目一共分为了7个模块，分别为：

block\_breaker模块：顶层模块，用于实例化

brick模块：砖块模块，控制砖块的显示，同时会返回小球的碰撞信息

controll模块：控制模块，用于控制挡板的移动

hit模块：碰撞模块，处理小球碰撞到边界或者碰撞到挡板的情况

movement模块：移动模块，控制小球移动，处理小球碰撞后的移动情况

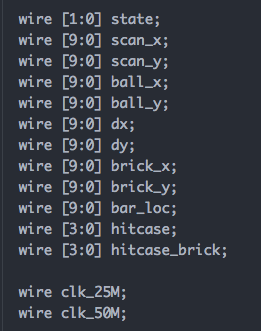
vga\_display模块：VGA显示模块，在这个模块里实例化ROM IP核

vga\_scan模块：VGA扫描模块

实现过程中我们将挡板的尺寸定义为64\*8

### block\_breaker模块

在block\_breaker模块中，主要任务是初始化信号，并且实例化所有的模块。主要的信号有：



state：游戏状态，00表示初始态，01表示正常状态，10表示暂停

scan\_x, scan\_y：当前扫描的点的坐标

ball\_x, ball\_y：小球的左上顶点坐标

dx, dy：小球的运动速度、方向

brick\_x, brick\_y：当前扫描到的砖块的左上顶点坐标

bar\_loc：挡板的左上顶点的坐标，以此确定挡板的位置

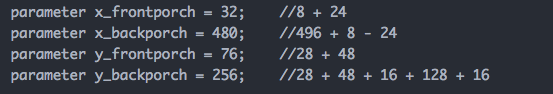
hitcase：小球碰撞到边界以及挡板后的状态

hitcase\_brick：小球碰撞到砖块后的信息

clk\_25M，clk\_50M：两个分频后分别为25MHz和50MHz的时钟

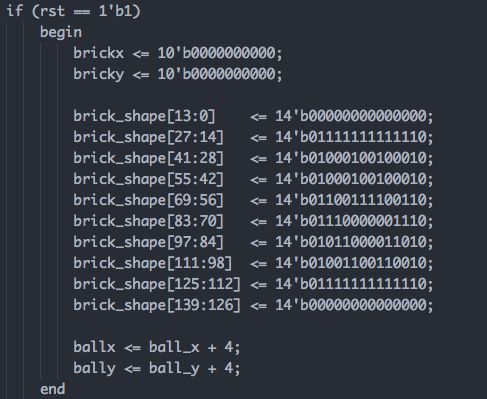
### brick模块

brick模块主要用于控制砖块的显示，并且处理小球碰撞砖块的情况。首先设置几个参数：



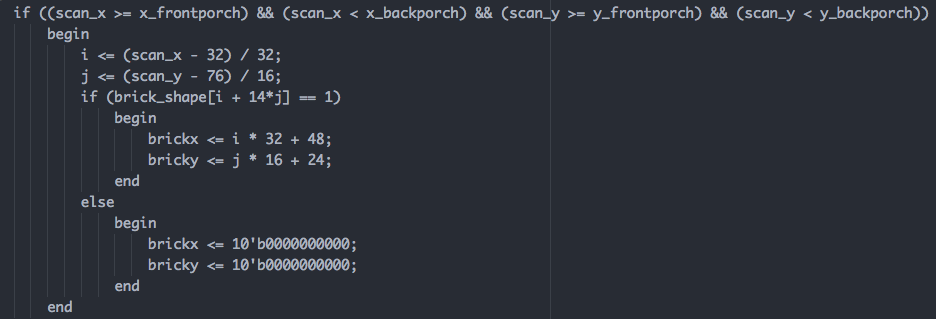
以此来划定砖块阵列的区域。

我们定义了一个140位的变量，每一位代表一块砖的显示状态，并进行初始化：



可以通过改变这张表来改变砖块的初始形态。

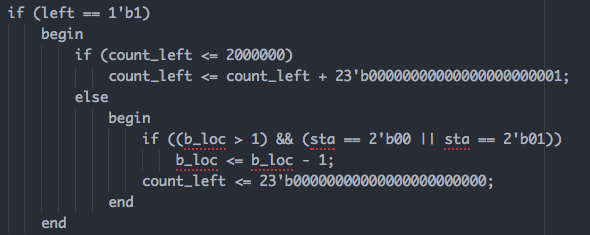
如果扫描的点在砖块阵列中，则需要进行砖块显示的判断



在brick模块中我们也处理了小球撞击砖块的情况，代码过长就不贴了，可去源码中查看。主要思想为：先判断小球的运动方向，以小球向左上方运动为例，共有三种可能，分别为撞击砖块的右侧，撞击砖块的下侧以及撞击砖块的右下角，分别就这三种可能做出判断并给hitcase\_brick赋值，传递到movement模块中对小球的运动进行改变。撞击相应的砖块后，对应的砖块的显示状态就需要置0。

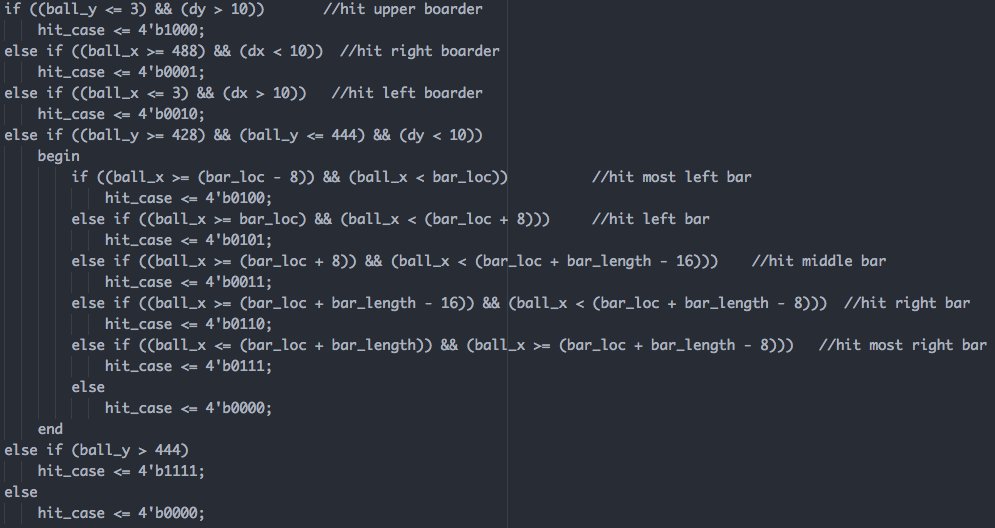
### controll模块

control模块用于控制挡板的运动，一共两个按键，检测到哪个按键后对挡板的位置变量bar\_loc进行相应改变即可。需要注意的是这里的按键需要做消抖操作，否则会出现一按按键挡板就会移动到最左边或者最右边的情况。已按下左移动键为例：



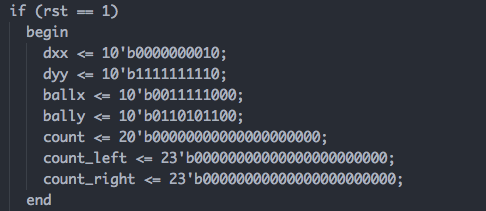
### hit模块

hit模块主要处理小球撞击边界以及挡板的情况，边界可参照上面的屏幕框架图来确认，挡板的位置则通过bar\_loc来确认。确认碰撞状态后将hitcase进行相应的赋值即可，小球之后的移动状态则在movement模块中进行改变。

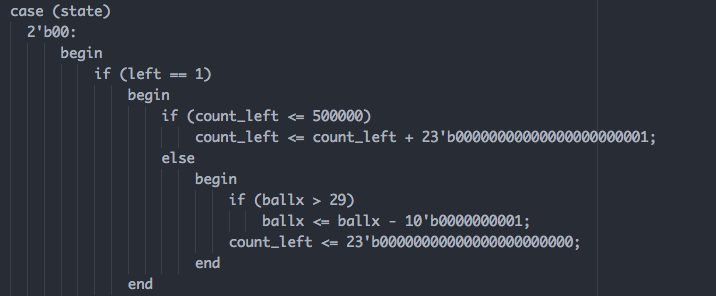


### movement模块

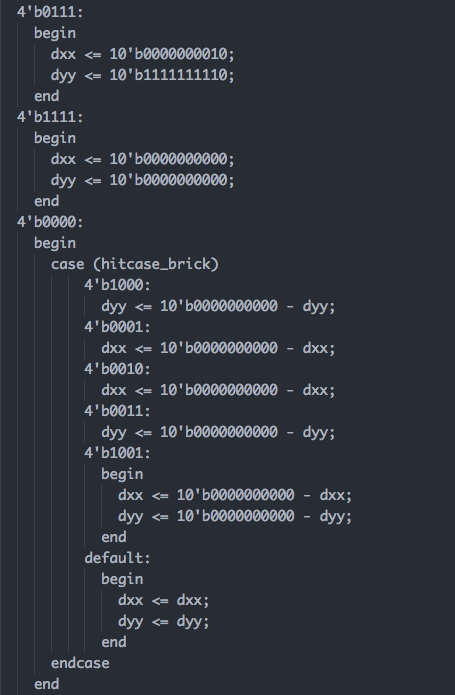
movement模块主要控制小球的运动，初始状态下我们将小球的运动方向设置右上方，小球的位置为挡板正中央。



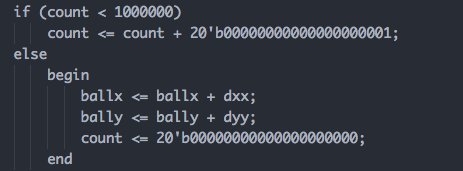
因为在游戏还没开始的时候，我们也可以控制挡板进行左右的移动来控制小球的发射位置，所以在movement模块中我们对state进行了判断，如果是初始状态，则小球也需要跟着移动，否则小球不移动。



在正常游戏状态下，我们需要对小球的碰撞情况进行判断，如果出现了碰撞，那么相应的运动方向需要进行改变

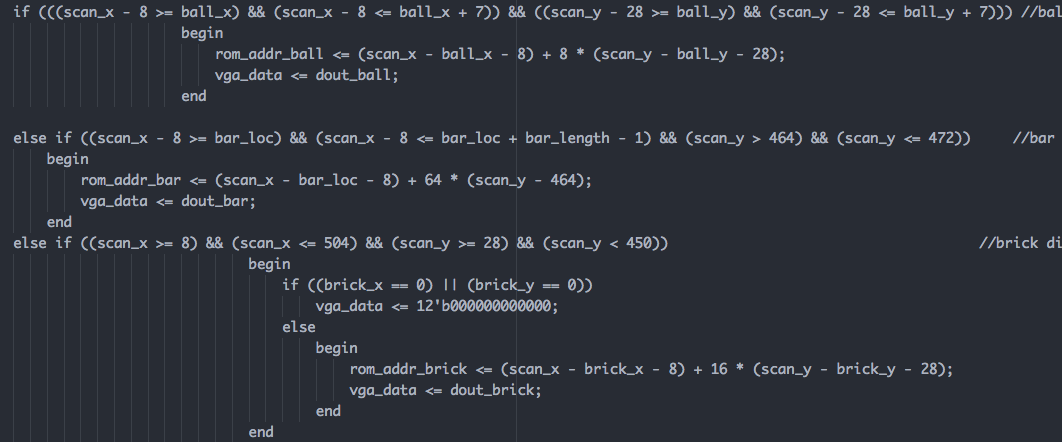


其中hitcase如果为1111，则表示挡板没有接到小球，小球停止运动，需要手动重置来重新开始游戏。为了不让小球移动地太快以至于肉眼看不到，我们设置一个count，将小球的运动速度放慢了100万倍，当然这是个可以调节的值，在以后的改动中争取将这个值设置为可由玩家调节。



### VGA显示模块和VGA扫描模块

两个和VGA有关的模块我们基本上就套用了老师的素材，对当前扫描的点进行判断，并根据别的模块传回来的参数进行显示即可。



需要注意的是在vga\_display模块中将ROM IP核实例化，否则无法显示图片。

## 实验结果

实验结果如下图：

初始状态：



打掉一部分砖块后：



1. **遇到的问题及解决方法**

设计以及实现过程中，我们也遇到了很多问题:

1. 遇到的最大的问题就是时钟问题，因为碰撞的状态存放在hitcase和hitcase\_brick中，hitcase和hitcase\_brick分别在hit模块和brick模块中赋值，但是在movement模块中处理，所以这几个模块就不能用同一个时钟，否则会出现延迟碰撞或者小球抖动的情况。最后我们在hit模块中使用了100MHz时钟，movement模块和brick模块使用了25MHz时钟，问题得到了解决。
2. 第二个问题就是消抖问题，本来想是否能够专门写一个消抖模块来实现消抖，但是考虑到我们一共只有两个按键需要消抖，小球的运动需要延时但是无法合并成一个，专门写两个模块费时费力，所以就整合到了各自的模块中，效果还不错。
3. 最后还有一个暂时没解决的问题，因为ROM IP中使用的coe文件都是a\*b的形式，只能显示矩形，无法显示圆形，所以我们的小球虽然在视觉上是圆形的，但是实际上是一个矩形，只是将非小球部分控制成了黑色和背景色一样，在碰撞过程中仔细看还是能看出周围的黑色部分的，但是时间有限，我们最终也没找到解决的办法，也算是一种缺憾吧。
4. **设计总结**

A：

本次实验我作为组长，压力还是挺大的。不过好在上学期已经学习了计算机硬件系统设计这门课，对Verilog语言、VGA等已经有了一个比较全面的了解，所以上手的过程比较快。

本次实验我参与了所有过程，从确定设计目标到最后完成。个人认为最困难的两个阶段是确定设计目标和最后的调试debug阶段，确定设计目标我们小组成员讨论了很久很久最后才最终确定做一个打砖块游戏。debug阶段不用说，每次一点点小的错误就需要改很久，最开始我们的小球无法反弹，沿着路线直接飞出了屏幕，过了一会儿又自行飞了回来，我们找了半天最后终于找到是边界条件设置的问题。修改之后小球能够正确反弹了，但是又出现了抖动的问题，于是又得一个模块一个模块的改代码。debug过程中还用到了模块三中的探针用于帮助调试，最后懒得删除了，所以源码中会有我们调试的部分代码。

总的来说这次实验我还是比较满意的，亲手做出了这么一个游戏，虽然可能还有很多问题，但是最后的成就感还是很大的。希望以后还有机会可以进一步改进吧。

B：

在小学期我们学习了计算机组成原理课程设计这门课程，它是一门综合应用计算机组成原理课程的知识和内容来解决问题并实现功能的课程，

通过学习这门课的知识并完成实验的过程，知悉和理解了现代接口技术，知悉了现代数字系统设计方法和基本流程，并学习一门硬件描述语言，掌握了常用接口硬件的功能、特点及使用方法，能够读懂简单的接口硬件原理图，根据要求设计接口控制模块。

并且运用这些知识，完成了简单I/O 控制，如LED、Switch 和七段数码管，VGA 控制器设计实验。设计一个 VGA 显示控制器，支持 640×480×60Hz。

课程知识的实用性很强，因此实验就显得非常重要，刚开始做实验的时候，由于自己的理论知识基础不好，在实验过程遇到了许多的难题，也使我感到理论知识的重要性。但是我并没有气垒，在实验中发现问题，独立思考，团队合作，最终解决问题，从而也就加深我对理论知识的理解。

C：

最开始我们遇到最大的难题还是选择设计目标，我最开始的意见是选择老师所给的题目，并没有想到更多的内容，后来经过小组讨论，最终确定了“打砖块”的题目，我分配到的任务是“消抖”，第一次接手这种硬件编程，遇到的困难还是蛮多的，遇到问题的时候也会比较焦急，实验时因为时间的限制，只能简单的学个大概，并没有更深层次的钻研，不过最后通过查阅资料和小组讨论的方式，还是顺利地得到了想要的结果。按键开关是各种电子设备不可或缺的人机接口。在实际应用中，很大一部分的按键是机械按键。在机械按键的触点闭合和断开时，都会产生抖动，为了保证系统能正确识别按键的开关，就必须对按键的抖动进行处理。经过查找资料，按键输出的信号的跳变时间（上升沿和下降沿）最大是在20ms左右。按键一次闭合最短的时间大概是120ms左右。如果我们把按键的输出做为一个时钟域（时钟频率未知，但信号的slow rate是已知的，既最大20ms左右）的信号，用另外一个时钟来采集这个按键的输出，则就可以把按键的消抖归结为一个最基本的CDC问题来处理。而问题的核心是如何确定采集时钟的频率。通过和组员的讨论，最终我们确定了具体的判断值。 花费最长时间的是看懂示例代码和明白抖动及消除抖动的原理，刚开始想的比较复杂，后来在组长的帮助下又进行了改进。