

湖南大学课程设计报告



基于 VHDL 语言的多人抢答器的设计与仿真分析

课程名称: 电子技术综合设计

学 院: 电气与信息工程学院

班 级: 自动化 1903 班

小组成员:

编号	姓名	学号
1	张宇熙	
2	张金昊男	
3	顾诗文	

2021 年 11 月 18 日

摘 要

本文基于 VHDL 语言和 Intel 公司的 QUARTUS 软件设计并仿真现实中生活中的知识抢答器系统，并在 DE2-115 款 FPGA 开发板上进行模拟功能验证。

针对参考文献与设计要求，本文提出两种设计思路与方案，一种是基于硬件的电路搭建与实体设计，一种是基于硬件描述语言的方案在相关平台上进行仿真实验后使用开发板验证。考虑到可操作性与成本的因素，本文考虑使用 VHDL 语言进行仿真与设计。

针对硬件描述语言的特点，考虑模块化设计，采用“自顶向下”的方式，针对每一个模块分别设计，最后通过软件，进行逻辑连线将所有模块串联起来。最后，本文针对鉴别模块，锁存模块，编码模块，计时模块，报警模块，译码模块等模块的原理与设计思路进行详细说明。最后，得到完整的抢答器系统的连线示意图与各个子模块的仿真波形。

针对上板验证，本文基于开发板的手册分配了合适的引脚与元件，将程序烧录到开发板中进行调试，最后得到符合实际情况的调试结果。

最后，小组成员针对此次仿真做了合适的总结与推广，感谢此次为小组电子技术综合设计提供帮助的老师和同学们。

关键词：VHDL 语言 多人抢答器系统 自顶向下 仿真模拟 开发板

目 录

基于 VHDL 语言的抢答器设计	1
一. 项目背景	1
二. 现行主流方法	1
三. 多人抢答器的设计思路与模块说明	1
四. 开发板与相关仿真软件简介	2
五. 抢答器系统的设计思路	3
5.1 抢答器系统的设计要求	3
5.2 系统的总体设计思想	3
六. 抢答器子系统的设计与实现	4
6.1 鉴别模块的设计与实现	4
6.2 锁存反馈模块的设计与实现	5
6.3 编码模块的设计与实现	6
6.4 报警模块的设计与实现	7
6.5 倒计时模块的设计与实现	8
6.6 组别译码显示模块的设计与实现	9
七. 抢答器系统的实现	9
7.1 模块的连接	9
7.2 开发板引脚的分配	10
7.3 程序烧录与开发板验证	11
八. 课程设计的体会与收获	13
8.1 团队的整体收获	13
8.2 团队合作情况	13
九. 附录: VHDL 程序	13

基于 VHDL 语言的抢答器设计

一. 项目背景

在知识竞赛、文体娱乐活动（抢答赛活动）中，能准确、公正、直观地判断出抢答者的座位号。更好的促进各个团体的竞争意识，让选手门体验到战场般的压力感。传统抢答器只是大概判断出抢答成功或犯规选手台号，无法显示出每个选手的抢答时间。而今抢答器可以通过数据来说明裁决结果的准确性、公平性。使比赛大大增加了娱乐性的同时，也更加公平、公正。

二. 现行主流方法

在目前市面上流行的抢答器中，有两大类方法实现抢答器的功能：一是使用集成单元电路实现相关的功能，另一种是使用硬件描述语言来实现相关的功能。这两种实现方法各有特色。我们下面对这两种方法的技术路线作一简单说明。

- 使用集成单元电路实现的技术路线：

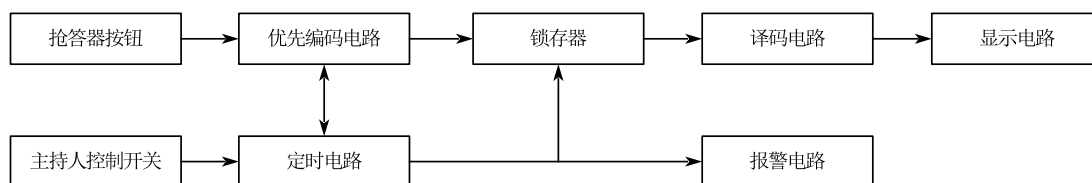


图 1 使用集成单元电路技术路线

- 使用硬件描述语言实现的技术路线：

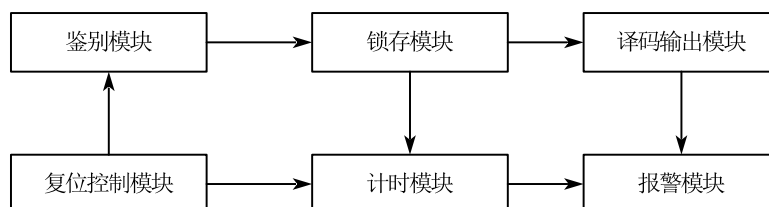


图 2 使用硬件描述语言实现的技术路线

从技术路线就可以明显地看出，这两种实现方案各不相同，显然路线一的硬件电路实现复杂，设计困难，一旦出错可能需要全盘推倒，因此，本次综合设计，我们采用第二种方案，使用硬件描述语言-VHDL 语言来实现抢答器的相关功能。采用硬件描述语言只需要一台电脑，在相关的仿真软件上做仿真，然后将调试程序烧录到 FPGA 开发板上即可实现，操作较方案一简单很多，实现起来成本也很低，可以反复修改。

三. 多人抢答器的设计思路与模块说明

（1）设计思路的简单阐述

根据对抢答器的具体要求，我们需要把待设计的抢答器分为具有以下功能模块：抢答信号的鉴别模块、编码模块、计时模块、译码显示模块、扬声器控制电路模块等。其中，显示模块中又包含抢答成功组别的显示电路模块和倒计时数值显示模块。这些模块封装在一起可以构成一个多人抢答器系统，烧录到开发板上即可实现简单的抢答器功能。

（2）相关模块说明

根据技术路线图二，我们将整个抢答器系统分为以下几个模块来分别实现：

1. 鉴别模块：该模块的功能是鉴别 8 组中是哪一组抢答成功并且把抢答成功的组别信

号输出给锁存模块。

2. 复位控制模块：我们将为主持人设置一个控制开关，这个开关构成一个模块，可以用来进行系统的清零和抢答的开始。
3. 锁存模块：该模块的作用是当第一个抢答者抢答后，对该抢答信号进行锁存，并且将其对应的组别显示在数码管上，后面的抢答者信号全都无响应，直到主持人按下复位键。
4. 报警模块：将各个模块的输入的不同信号经过译码变成 BCD 码之后在开发板的数码管上直接显示，有条件的还可以追加蜂鸣器的声音。
5. 计时模块：抢答成功后，主持人给出使能信号，抢答成员需要在规定的时间内给出答案，超时则会触发报警信号。
6. 译码输出模块：该模块的功能是将抢答信息编码成 4 位二进制代码，使得主持人能够判断是哪一组抢答成功。

四. 开发板与相关仿真软件简介

【1】 DE2-115 开发板简介

DE2-115 开发板拥有能够为用户实现广泛设计的特征，包括从简单的设计到各种多媒体电路的设计。

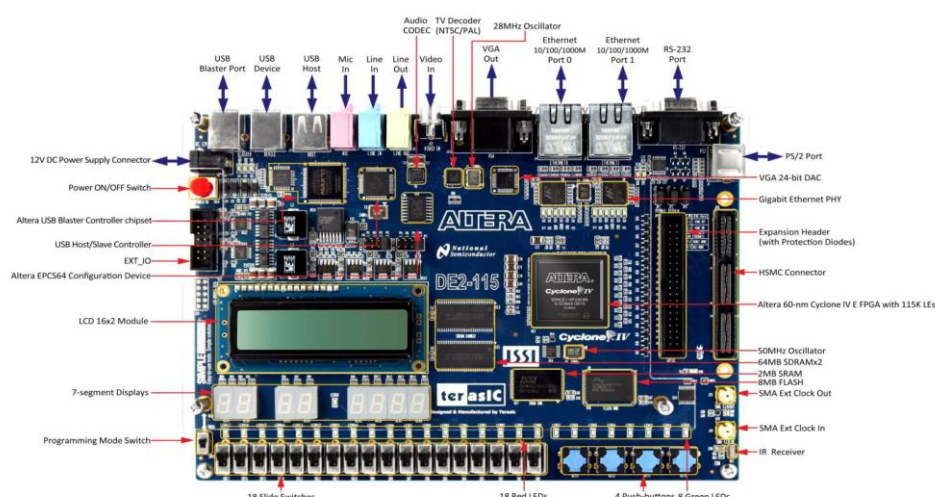


图 3 DE2-115 开发板(正面)

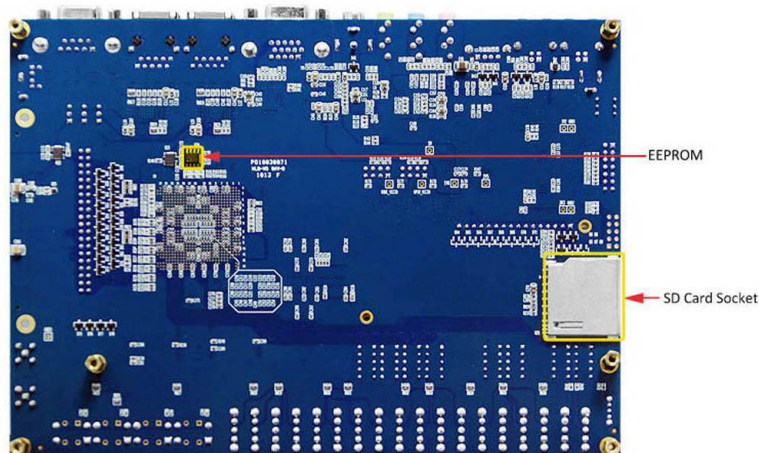


图 4 DE2-115 开发板(背面)

DE2-115 开发板包括以下硬件资源：

Altera Cyclone® IV 4CE115 FPGA 芯片、Altera 串行配置芯片—EPCS64、板上 USB Blaster 下载电路，同时支持 JTAG 模式和 AS 模式、2MB SRAM 、2片 64MB SDRAM 、8MB 闪存、SD 卡插槽、4 个按钮开关、18 个滑动开关、18 个红色 LEDs、9 个绿色 LED 等部件。

除了这些硬件功能外，DE2-115 开发板还支持标准 I/O 接口和用于评估各项组件的控制面板等软件工具。该软件也提供用于验证 DE2-115 开发板高级功能的大量实例演示。

【2】 QUARTUS II 软件简介

Quartus II 是 Altera 公司推出的综合性 CPLD/FPGA 开发软件，软件支持原理图、VHDL、VerilogHDL 以及 AHDL (Altera Hardware Support Description Language) 等多种设计输入形式，内嵌自有的综合器以及仿真器，可以完成从设计输入到硬件配置的完整 PLD 设计流程。

Quartus II 可以在 Windows、Linux 以及 Unix 上使用，除了可以使用 Tcl 脚本完成设计流程外，提供了完善的用户图形界面设计方式。具有运行速度快，界面统一，功能集中，易学易用等特点。Quartus II 支持 Altera 的 IP 核，包含了 LPM/MegaFunction 宏功能模块库，使用户可以充分利用成熟的模块，简化了设计的复杂性、加快了设计速度。对第三方 EDA 工具的良好支持也使用户可以在设计流程的各个阶段使用熟悉的第三方 EDA 工具。此外，Quartus II 通过和 DSP Builder 工具与 Matlab/Simulink 相结合，可以方便地实现各种 DSP 应用系统；支持 Altera 的片上可编程系统（SOPC）开发，集系统级设计、嵌入式软件开发、可编程逻辑设计于一体，是一种综合性的开发平台。

五. 抢答器系统的设计思路

5.1 抢答器系统的设计要求

一般来说，设计一台数字系统抢答器，必须要能够判断出第一位抢答者，并且通过数码管显示，蜂鸣器等途径能够让观众判断谁是成功抢答者，并设置一定的答题时间系统，使得抢答者只能在规定的时间内抢答，主持人根据答题结果控制抢答器的清零与复位，掌握比赛进程，因此，根据具体要求，我们设计出的抢答器具体要求如下：

- (1) 抢答器容纳 8 组选手，每一组选手有一个按钮可供抢答使用，主持人有一个按钮，用来控制系统的清零(组别显示的数码管熄灭)，进行下一轮次的抢答；
- (2) 电路能够对第一组成功抢答的信号进行锁存，鉴别和显示，在主持人将系统复位并发出抢答指令后，若选手按下抢答按钮，则该组别的信号则被锁存，并在数码管上显示组别，蜂鸣器给出声音提示，同时，电路具备自锁功能，此时其他组别的抢答按钮不起作用；
- (3) 抢答器具有限时答题的功能，当主持人启动倒计时按钮后，要求计时器采用倒计时，同时最后计时器到 00 时蜂鸣器会给出声音提示。

5.2 系统的总体设计思想

我们根据要求进行系统的设计，熟悉了每一模块的功能后，我们便可以开始进行设计，对于各个模块，我们的设计思路如下：

抢答的过程：

1. 主持人按下复位键(CLR)，此时系统进入抢答状态，计时模块输出初始信号给数码管模块并显示初始值；

2. 当某个参赛组抢先按下抢答按钮时，系统将其余组别的抢答按钮封锁，同时，蜂鸣器发出声音提示，组别显示模块送出信号给数码管模块，显示抢答成功的小组编号并保持到主持人清零；
3. 主持人对抢答信号进行确认，计时模块送出倒计时允许信号，计时显示器开始倒计时，若选手超时则蜂鸣器鸣叫，若未超时，则主持人可以按下停止键，放置蜂鸣器鸣叫；
4. 主持人按下复位键(CLR),即可清空本轮抢答成功组别，开始进行下一轮次的抢答。

此抢答器的设计可以使用自顶向下的设计路线，使用 VHDL 硬件语言对各个模块进行层次化，系统化的描述，先设计一个顶层文件，再将各个子系统连接起来形成有机的整体。

六. 抢答器子系统的设计与实现

6.1 鉴别模块的设计与实现

鉴别模块用以准确直观地判断 S1,S2,S3,S4,S5,S6,S7,S8,这几组抢答者谁最先按下按钮，并为显示端送出信号，通过数显和蜂鸣等途径，观众能够清楚地知道是哪一组抢答成功，这是整个鉴别模块的核心部分。同时，组别显示端为下一模块送出信号，以便主持人进行下一步的操作。图 5 给出了抢答鉴别模块元件的引脚示意图。

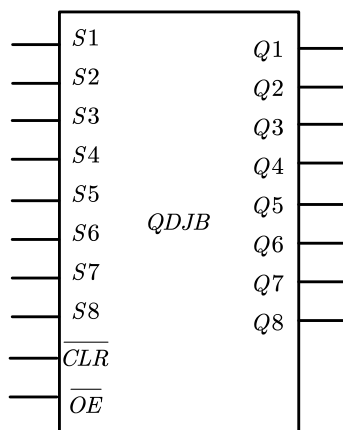


图 5 抢答鉴别元件示意图

引脚的作用与说明：

输入信号：各组的抢答按钮 S1~S8，系统的清零按钮：CLR，反馈使能端：OE；

输出信号：各组的抢答按钮显示端：Q1~Q8

工作原理：第一个按下键的小组，抢答信号判定模块 QDJB 通过缓冲输出信号的反馈将本轮参赛组抢先抢答的信号锁存，并且通过异步清零的方式将其他参赛信号组的信号屏蔽，显示组别直到主持人发出复位信号为止。当 CLR=1 时系统复位，抢答被屏蔽，当 CLR=0 时，且 OE=0 时，使其进入抢答鉴别状态。图 6 展示了该模块的功能。

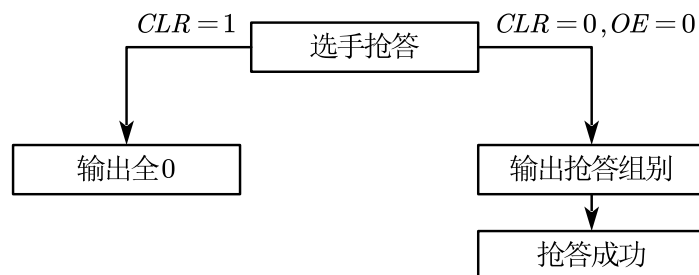


图 6 模块功能

我们在 QUARTUS 软件中进行仿真，得到的仿真波形如图 7 所示。

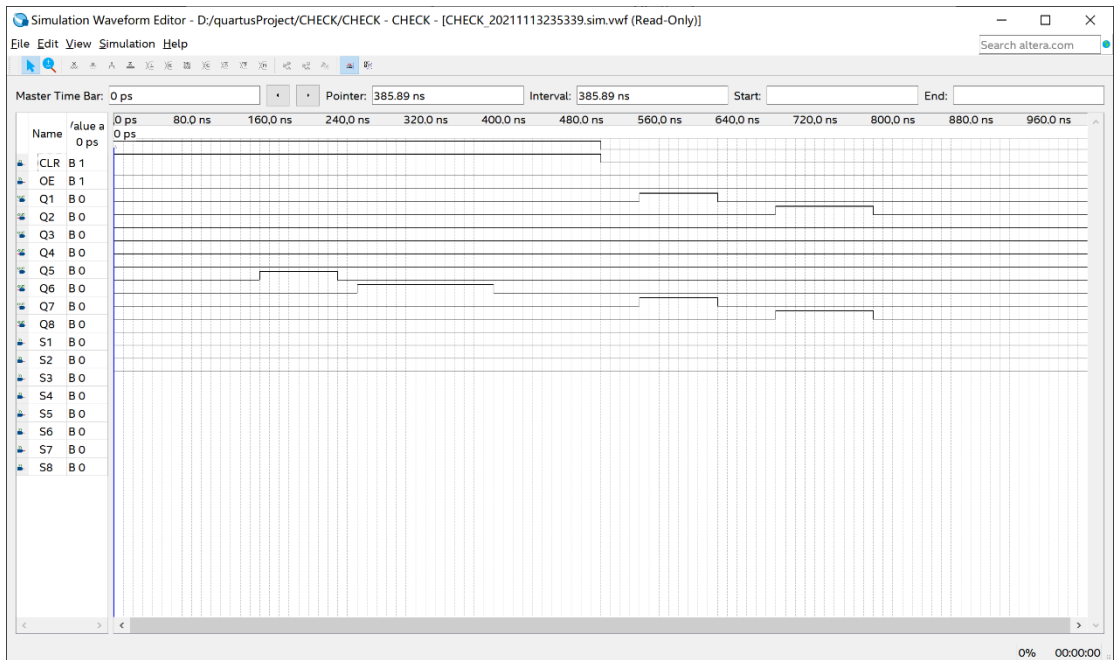


图 7 鉴别模块仿真波形图

6.2 锁存反馈模块的设计与实现

当抢答鉴别模块成功将各个抢答组的抢答信号输出后，必须由锁存电路来将抢答信号中最先抢答的组别锁定，禁止其他组的信号显示出来，这个模块是整个抢答器系统最重要的部分，它的性能将直接影响主持人对于比赛公平进行的判断。锁存模块的元件的示意图如图 8 所示。

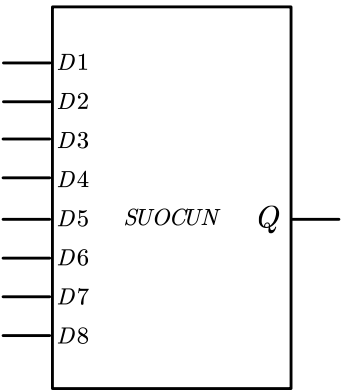


图 8 锁存鉴别模块示意图

引脚的作用与说明：

输入信号：D1~D8 是各组的抢答信号。

输出信号：Q 是锁存反馈信号。

工作原理：当 D1~D8 各组的抢答信号进入锁存模块时，在锁存模块中，对输入的各个信号的信息进行或逻辑运算，将计算结果输出给 Q，通过 Q 向外输出，最后将 Q 值反馈给 QDJB 模块，对本次的抢答结果进行锁存，将锁存结果输出给下一级电路。我们在 QUARTUS 软件中进行仿真，得到的仿真波形如图 9 所示。

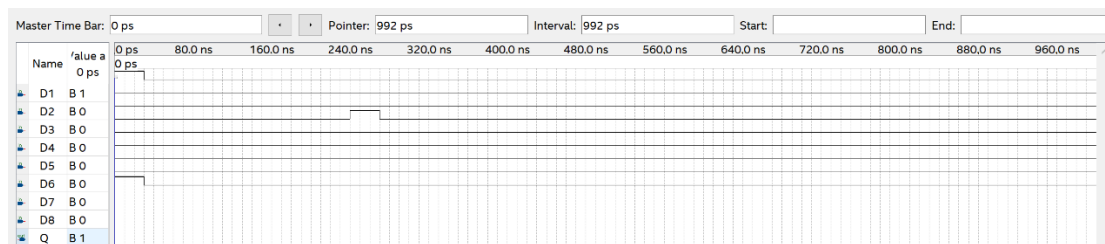


图 9 锁存反馈模块仿真波形图

6.3 编码模块的设计与实现

当被锁存的信号被送到编码模块后，该模块会将送到的各组的信息进行编码，该模块在总体电路中起到对信息编码的作用，以便在后级的译码电路中正确显示，编码模块的元件示意图如图 10 所示。

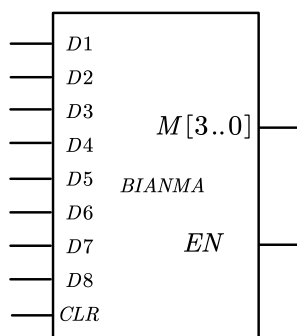


图 10 编码模块示意图

引脚的作用与说明：

输入信号：锁存后，信号的输入端：Q1~Q8,模块的清零端为 CLR。

输出信号：编码后的输出端为 M[3..0],蜂鸣器响起的使能端为 EN。

工作原理：通过编程的方式实现编码功能。

我们通过编程，在 QUARTUS 软件中给出仿真波形如图 11 所示。

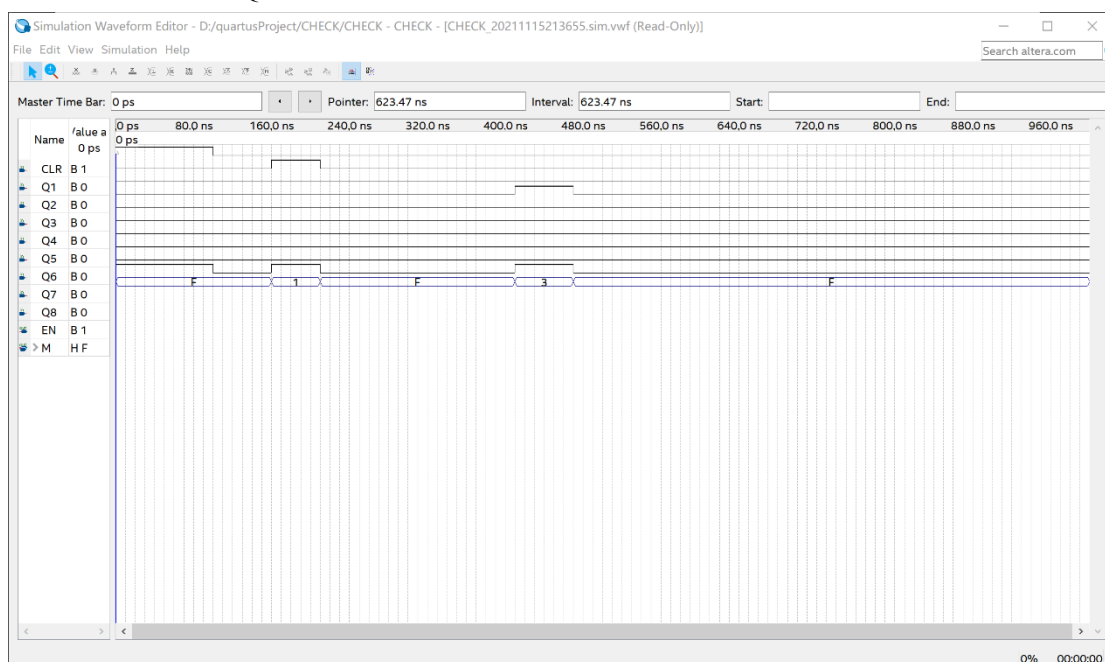


图 11 编码模块仿真波形示意图

6.4 报警模块的设计与实现

当其中的某一组抢答成功之后，为了能让主持人第一时间反应到抢答成功，系统需要设置一个声音报警器，来提示主持人进行后续步骤，该模块在系统中是十分必要的，该模块的设计能够为比赛顺利进行提供保障，该模块的元件示意图如图 12 所示。

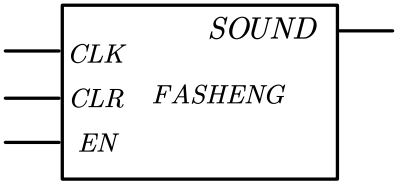


图 12 报警模块示意图

引脚的作用与说明：

输入信号：时钟信号 CLK,复位端 CLR，声音响起的使能端 EN。

输出信号：声音响起的输出端：SOUND。

工作原理：当时钟脉冲的上升沿到来时，使能端 EN 为高电平，CLR 端的信号为低电平时，SOUND 端输出高电平，此时声音响起；当 CLR 为高电平时屏蔽一切 EN 端的信号，SOUND 端输出低电平，声音不响起。原理示意图如图 13 所示。

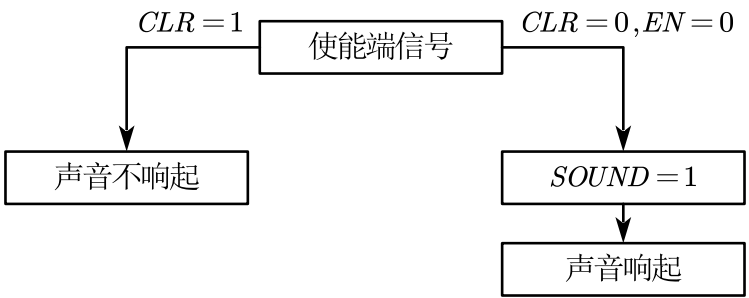


图 13 报警模块原理示意图

我们在 QUARTUS 软件中仿真该模块的波形图如图 14 所示。

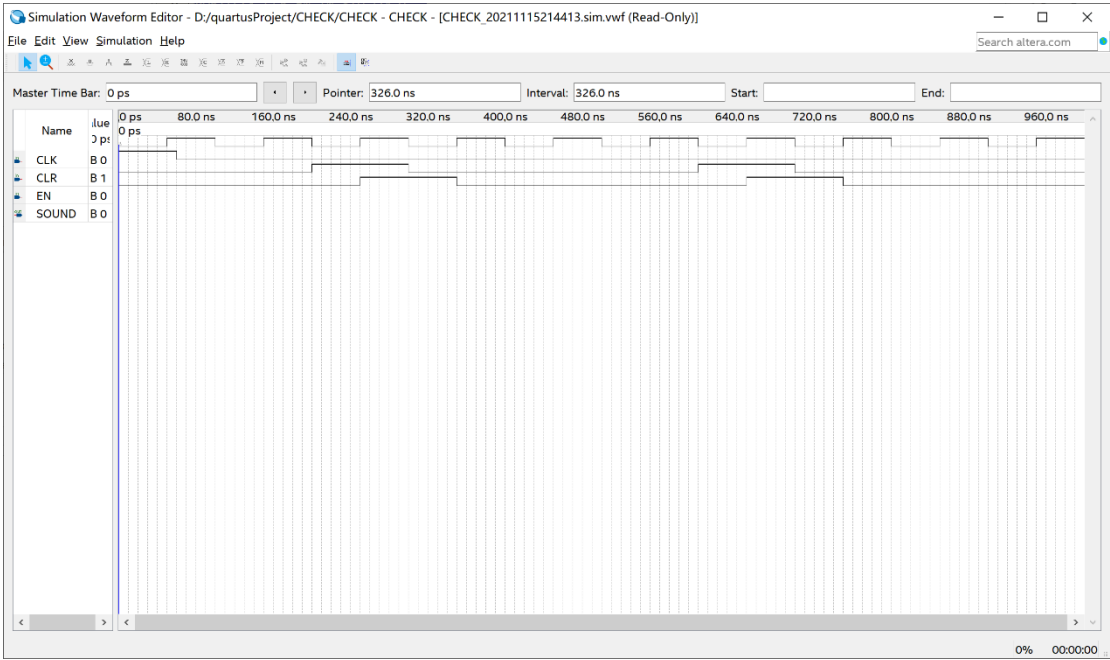


图 14 报警模块仿真波形

6.5 倒计时模块的设计与实现

当其中的某组抢答成功之后,需要由该组的成员来进行回答,这一进程需要有时间控制,答题进程的开始与结束需要由主持人来掌握,倒计时时间由数码管显示出来提示观众和主持人。该模块的元件示意图如图 15 所示。

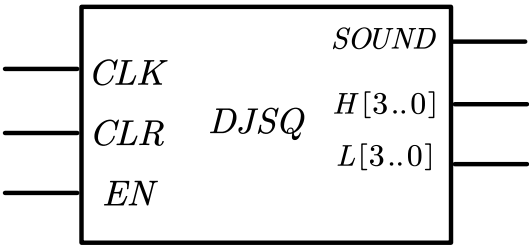


图 15 倒计时模块元件示意图

引脚的作用与说明:

输入信号: 时钟信号 CLK, 复位端 CLR, 倒计时使能端 EN;

输出信号: 倒计时高位显示端 H[3..0],倒计时低位显示端 L[3..0],倒计时结束提示音输出端 SOUND。

工作原理: 如果一组抢答成功,主持人给出判断并给出使能信号使倒计时开始计时如 果计 时到 30 秒的时候声音就会响起给主持人提供信息说明抢答队员已经超过了规定的时 间, 主持人会根据自己的主观意愿宣布此次抢答有效或无效。总的来说倒计时起到提醒参 赛者 时间的结束,并起到给主持人提示的作用。倒计时的设计思想: 倒计时需要用两个数码 管 显示,其中一个数码管显示十位(H)另一个显示个位(L),它们都用二进制表示,当 参赛 者抢答成功时,主持人给出是否有效,如果有效就把使能信号 EN 赋低电平,倒计时 开始 工作,当时钟脉冲(CLK)有效的时候倒计时就开始计时,当个位变成零的时候程序 就会 使十位减 1 并且个位变成 9,如果个位没有变成零的时候个位在时钟脉冲上升沿的时 候自动 减 1,十位保持不变。当倒计时结束时,会有声音响起,提醒回答者和主持人回答 超时,主 持人对倒计时进行复位,回答结束。原理框图如图 16 所示:

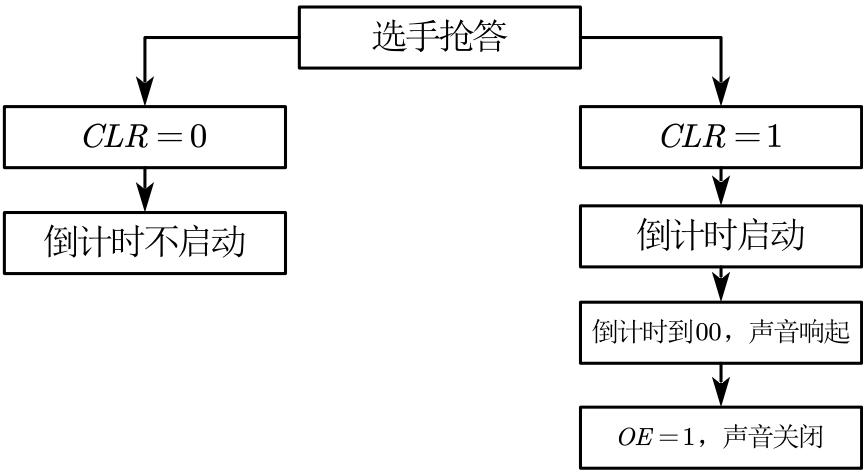


图 16 倒计时模块原理示意图

我们在 QUARTUS 软件中对该模块进行仿真得到的仿真波形如图 17 所示:

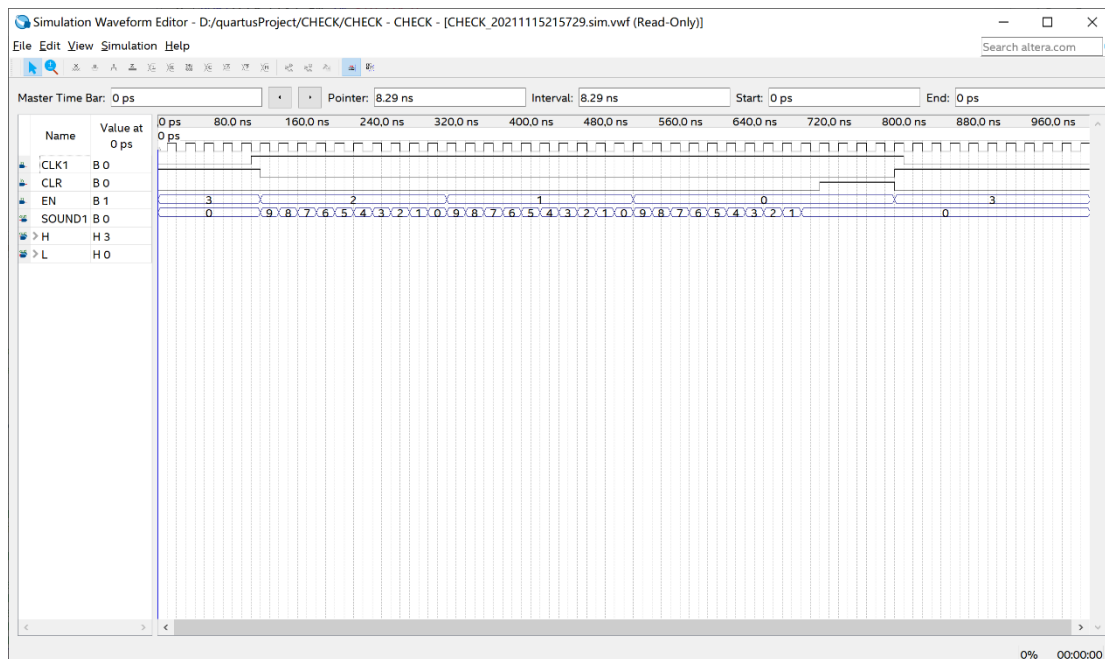


图 17 倒计时模块仿真波形

6.6 组别译码显示模块的设计与实现

当某一小组抢答成功之后，编码模块将抢答信息编码成四位二进制数，传送给译码显示模块，而译码显示模块的功能就是将四位二进制数通过数码管显示出来，以便于观众和主持人能够判断具体是哪一组抢答成功。该模块元件的示意图如图 18 所示。

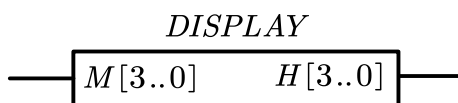


图 18 组别译码模块元件示意图

引脚的作用与说明：

输入信号：编码信号的输入端 $M[3..0]$;

输出信号：数码管显示输出端 $BCD[7..0]$;

工作原理：译码显示模块，将四位二进制 BCD 码转换为七段二进制数字，用阿拉伯数字的形式输出在数码管上，数码管由七段组成，根据各个模块的信号决定这七段那些显示而哪些不显示生成抢答组别的组号数字。在仿真与设计的过程中，要求确定我们最终所使用的开发板上的数码管是共阴极还是共阳极！

七. 抢答器系统的实现

7.1 模块的连接

我们在经过讨论与设计，实现了抢答器各个子模块的功能与仿真分析。我们还需要一根主线将所有的模块连接起来构成一个有机整体，才能实现抢答器的工作。这就是自顶向下设计中的顶层文件。我们可以使用 VHDL 语言进行编程进行各个模块的控制，调用与连接，对各个元件进行连线。QUARTUS 软件为开发者开发了顶层元件原理图的功能，设计者可以通过该功能实现元件的连接，清晰地看到模块连接的原理。

图 20 展示了我们对于抢答器各个模块进行连接控制的顶层元件线路图。

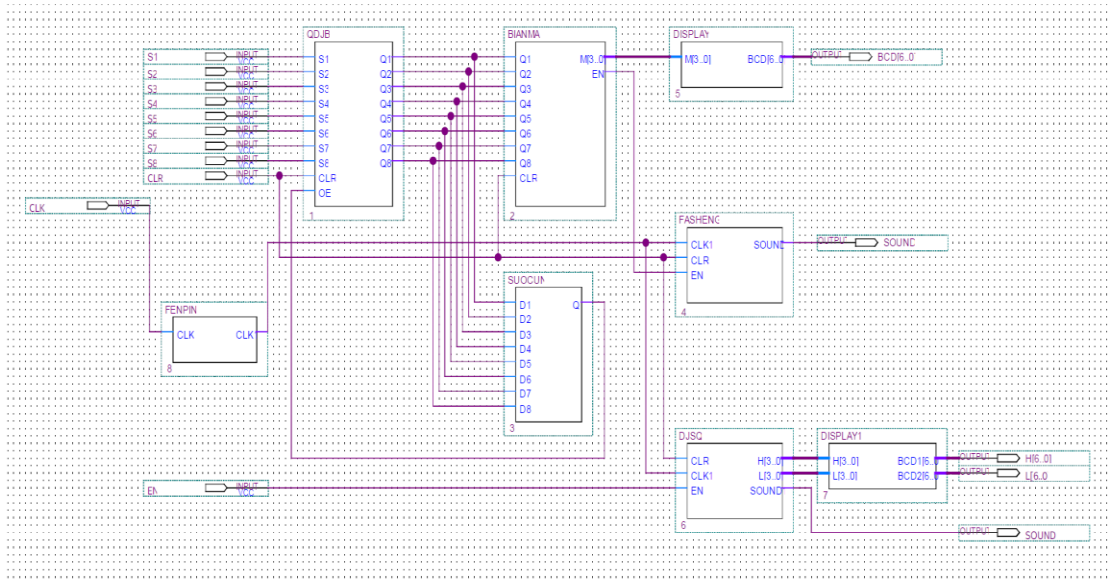


图 20 顶层连线示意图

在 QUARTUS 软件中抢答器的整体仿真波形如图 21 所示。

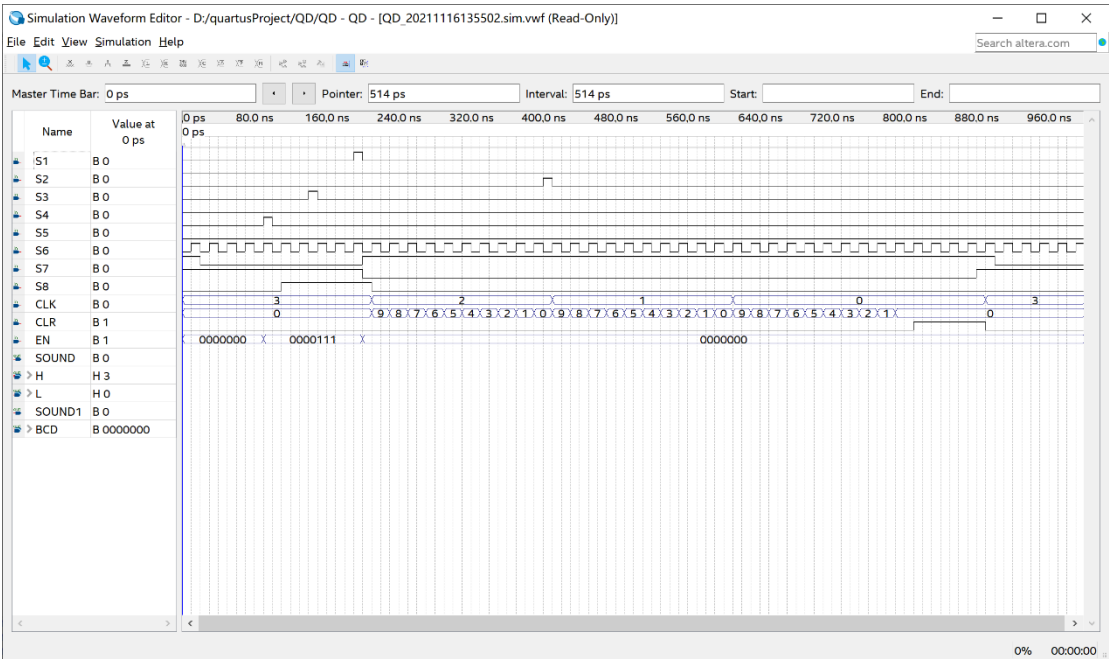


图 21 抢答器系统波形仿真

7.2 开发板引脚的分配

我们在软件上完成一系列的仿真之后，需要在开发板上验证相关的功能。DE2-115 开发板提供了许多的机械按钮以及数码管，LED 灯，这些需要我们去进行分配后才能使用。我们在 QUARTUS 软件中的引脚分配界面如图 22 所示。

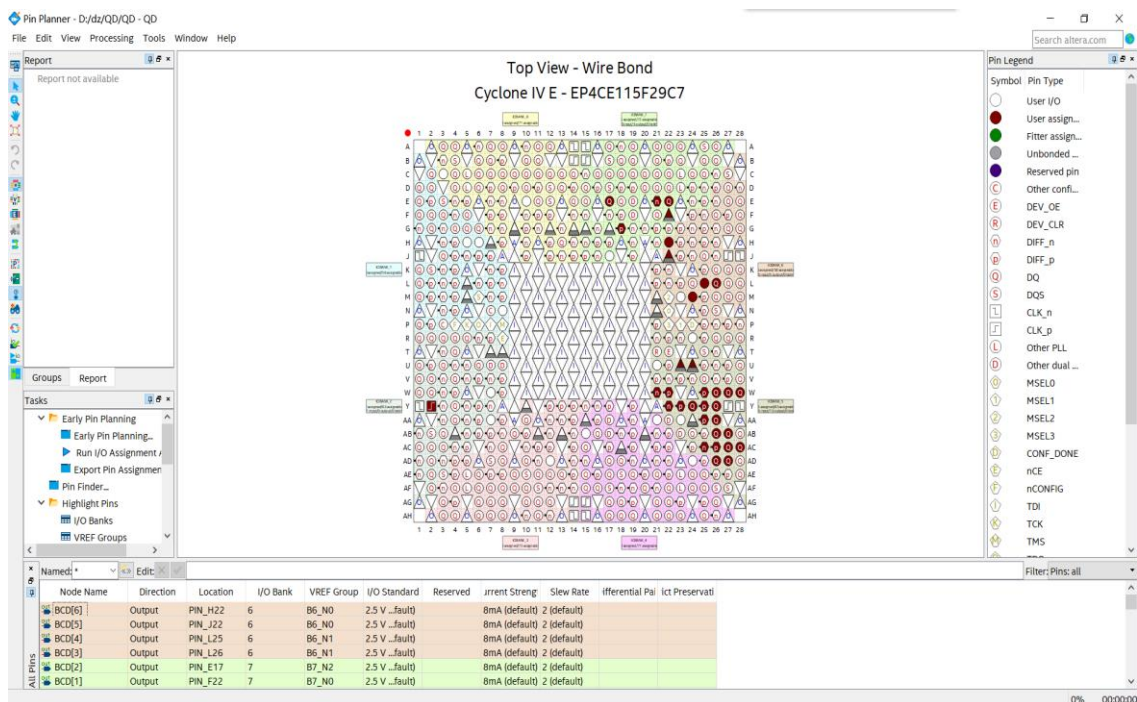


图 22 引脚分配界面

具体的引脚，数码管，LED(代替蜂鸣器)，机械按键的选择与分配如图 23 所示。































Named: *		Edit:  									
	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	rrrent Streng	Slew Rate	ifferential Pai	ict Preservati
	BCD[6]	Output	PIN_H22	6	B6_N0	2.5 V ...fault		8mA (default)	2 (default)		
	BCD[5]	Output	PIN_J22	6	B6_N0	2.5 V ...fault		8mA (default)	2 (default)		
	BCD[4]	Output	PIN_L25	6	B6_N1	2.5 V ...fault		8mA (default)	2 (default)		
	BCD[3]	Output	PIN_L26	6	B6_N1	2.5 V ...fault		8mA (default)	2 (default)		
	BCD[2]	Output	PIN_E17	7	B7_N2	2.5 V ...fault		8mA (default)	2 (default)		
	BCD[1]	Output	PIN_F22	7	B7_N0	2.5 V ...fault		8mA (default)	2 (default)		
	BCD[0]	Output	PIN_G18	7	B7_N2	2.5 V ...fault		8mA (default)	2 (default)		
	CLK	Input	PIN_Y2	2	B2_N0	2.5 V ...fault		8mA (default)			
	CLR	Input	PIN_Y23	5	B5_N2	2.5 V ...fault		8mA (default)			
	EN	Input	PIN_Y24	5	B5_N2	2.5 V ...fault		8mA (default)			
	H[6]	Output	PIN_W28	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	H[5]	Output	PIN_W27	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	H[4]	Output	PIN_Y26	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	H[3]	Output	PIN_W26	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	H[2]	Output	PIN_Y25	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	H[1]	Output	PIN_AA26	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	H[0]	Output	PIN_AA25	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	L[6]	Output	PIN_U24	5	B5_N0	2.5 V ...fault		8mA (default)	2 (default)		
	L[5]	Output	PIN_U23	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	L[4]	Output	PIN_W25	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	L[3]	Output	PIN_W22	5	B5_N0	2.5 V ...fault		8mA (default)	2 (default)		
	L[2]	Output	PIN_W21	5	B5_N1	2.5 V ...fault		8mA (default)	2 (default)		
	L[1]	Output	PIN_Y22	5	B5_N0	2.5 V ...fault		8mA (default)	2 (default)		
	L[0]	Output	PIN_M24	6	B6_N2	2.5 V ...fault		8mA (default)	2 (default)		
	S1	Input	PIN_AC25	5	B5_N2	2.5 V ...fault		8mA (default)			
	S2	Input	PIN_AC28	5	B5_N2	2.5 V ...fault		8mA (default)			
	S3	Input	PIN_AC27	5	B5_N2	2.5 V ...fault		8mA (default)			
	S4	Input	PIN_AD27	5	B5_N2	2.5 V ...fault		8mA (default)			

图 23 按键，LED，数码管的选择与分配

7.3 程序烧录与开发板验证

分配完引脚过后，我们将程序烧录到开发板上，在 QUARTUS 软件中，给出了烧录接口，其烧录界面如图 24 所示。

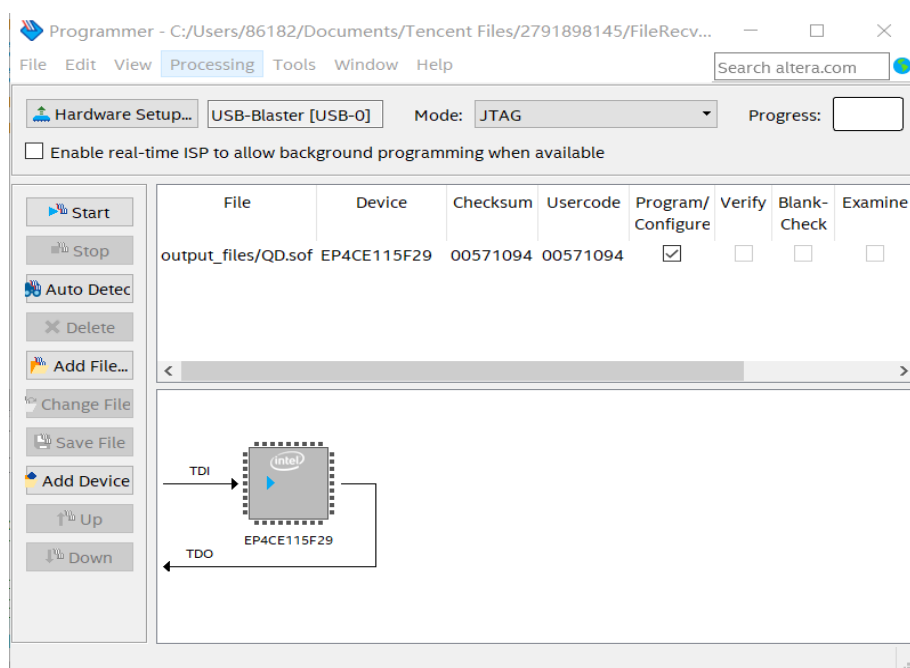


图 24 烧录界面

我们在开发板上验证功能如图 25~27 所示。

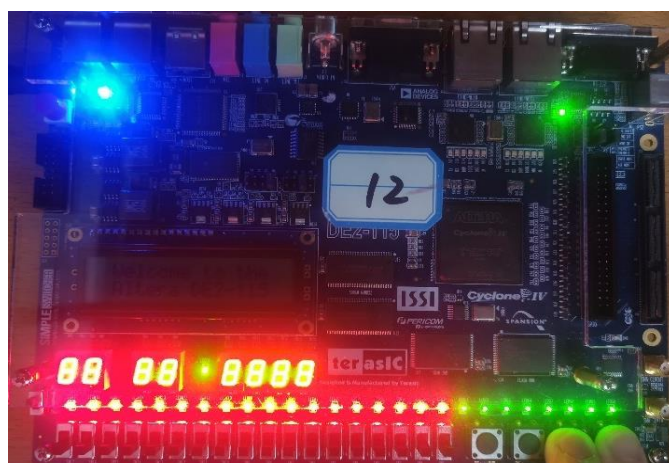


图 25 开发板初始状态

主持人第一轮复位，按下 CLR 按键，数码管显示倒计时初始状态 30 秒，并保持。如图 26(a) 所示。



图 26(a) 按下 CLR 按键

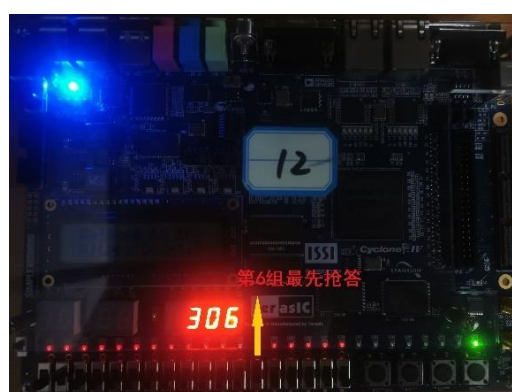


图 26(b) 第 6 组抢答成功

假设第六组最先抢答,则所存模块将其锁存,数码管会显示数字 6,蜂鸣器会响起(如图中绿色 LED 所示)如图 26(b)所示。
主持人根据场上的情况按下倒计时的使能端,使得倒计时开启如图 27(a)所示,倒计时结束后会触发蜂鸣器响起(绿色 LED 亮起),如图 27(b)所示。



图 27(a) 倒计时进行中

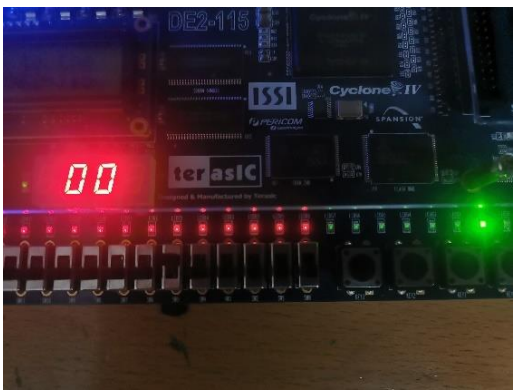


图 27(b) 倒计时结束

经过调试,我们认为我们的电子综合设计的成果基本达到了任务设计的要求,各项任务进程良好,体验指标正常,基本完成了此次的设计。

八. 课程设计的体会与收获

8.1 团队的整体收获

从一开始的组队,选题再到最后做仿真设计,设计模块的功能,编写程序代码,我们小组在这个项目上花费了较多的时间,最终还是得到了较为满意的结果。在这期间,我们不仅收获了友谊,还有难得的团队协作力。晚上在寝室,我们一点一点地查找错误,一起阅读开发板手册,最终找到原来是共阴极与共阳极的问题导致译码电路异常最终解决了问题使得数码管显示正常。从书本到实践,我们向前迈了一条腿,将理论知识运用到实践操作,既锻炼了我们的动手能力也考察了我们的实际运用能力,在今后的人生发展中会大有裨益。

8.2 团队合作情况

首先是查阅相关文献,确定课题与功能,这一部分的前期工作由三人共同商讨确定。
然后是软件部分,几部分的子模块设计由我们三人(张宇熙,张金昊男,顾诗文)分工,每个人负责几块,最后汇总到张金昊男同学那里,由他负责整体的调试,设计顶层逻辑连接文件,并对每一模块进行测试,给出仿真波形。
最后是报告的撰写,是由张宇熙同学撰写完成,顾诗文同学校对检验,我们三人共同阅读完成终稿,并打印成纸质版。

九. 附录: VHDL 程序

Code 1:QDJB 功能: 抢答鉴别模块程序源代码
LIBRARY IEEE; USE IEEE.STD_LOGIC_1164.ALL; USE IEEE.STD_LOGIC_UNSIGNED.ALL; ENTITY QDJB IS

```

PORT(S1,S2,S3,S4,S5,S6,S7,S8,CLR,OE:IN STD_LOGIC;
      Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8:OUT STD_LOGIC);
END QDJB;
ARCHITECTURE A OF QDJB IS
    BEGIN
PROCESS(S1,S2,S3,S4,S5,S6,S7,S8,OE,CLR)
    BEGIN
IF(CLR='1')THEN
Q1<='0';Q2<='0';Q3<='0';Q4<='0';Q5<='0';Q6<='0';Q7<='0';Q8<='0';
ELSIF(OE='0')THEN
Q1<=S1;Q2<=S2;Q3<=S3;Q4<=S4;Q5<=S5;Q6<=S6;Q7<=S7;Q8<=S8;
END IF;
END PROCESS;
END A;

```

Code 2:BIANMA

功能： 编码模块程序源代码

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY BIANMA IS
PORT(Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8:IN STD_LOGIC;
      CLR:IN STD_LOGIC;
      M:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      EN:OUT STD_LOGIC);
END BIANMA;
ARCHITECTURE A OF BIANMA IS
    BEGIN
PROCESS(Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,CLR)
    VARIABLE TEMP: STD_LOGIC_VECTOR(7 DOWNTO 0);
    BEGIN
TEMP:=Q1&Q2&Q3&Q4&Q5&Q6&Q7&Q8;
CASE TEMP IS
WHEN"10000000"=>M<="0001";
WHEN"01000000"=>M<="0010";
WHEN"00100000"=>M<="0011";
WHEN"00010000"=>M<="0100";
WHEN"00001000"=>M<="0101";
WHEN"00000100"=>M<="0110";
WHEN"00000010"=>M<="0111";
WHEN"00000001"=>M<="1000";
WHEN OTHERS=>M<="1111";
END CASE;
EN <=TEMP(7) OR TEMP(6) OR TEMP(5) OR TEMP(4) OR TEMP(3) OR TEMP(2) OR

```

```
TEMP(1) OR TEMP(0) OR CLR;
END PROCESS;
END A;
```

Code 3:SUOCUN

功能：锁存模块程序源代码

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY SUOCUN IS
PORT(D1,D2,D3,D4,D5,D6,D7,D8:IN STD_LOGIC;
      Q:OUT STD_LOGIC);
END SUOCUN;
ARCHITECTURE A OF SUOCUN IS
BEGIN
PROCESS(D1,D2,D3)
BEGIN
IF D1='1' OR D2='1' OR D3='1' OR D4='1' OR D5='1' OR D6='1' OR D7='1' OR D8='1' THEN
Q<='1';
ELSE Q<='0';
END IF;
END PROCESS;
END A;
```

Code 4:FASHENG

功能：报警模块程序源代码

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY FASHENG IS
PORT(CLK1,CLR,EN:IN STD_LOGIC;
      SOUND:OUT STD_LOGIC);
END FASHENG;
ARCHITECTURE A OF FASHENG IS
BEGIN
PROCESS(EN,CLK1)
BEGIN
IF(CLK1'EVENT AND CLK1='1') THEN
IF(CLR='0' AND EN='1') THEN
SOUND<='1';
ELSE
SOUND<='0';
END IF;
END IF;
```

```

END IF;
END PROCESS;
END A;

```

Code 5:DISPLAY

功能：译码模块第一段程序源代码

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DISPLAY IS
    PORT(M:IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         BCD:OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
END DISPLAY;
ARCHITECTURE A OF DISPLAY IS
    BEGIN
PROCESS(M)
BEGIN
    CASE M IS
        WHEN "0000"=>BCD<="1000000";
        WHEN "0001"=>BCD<="1111001";
        WHEN "0010"=>BCD<="0100100";
        WHEN "0011"=>BCD<="0110000";
        WHEN "0100"=>BCD<="0011001";
        WHEN "0101"=>BCD<="0010010";
        WHEN "0110"=>BCD<="0000010";
        WHEN "0111"=>BCD<="1111000";
        WHEN "1000"=>BCD<="0000000";
        WHEN "1001"=>BCD<="0010000";
        WHEN OTHERS => BCD <="1111111";
    END CASE;
END PROCESS;
END A;

```

Code 6:DJSQ

功能：倒计时模块程序源代码

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DJSQ IS
PORT(CLR,CLK1,EN:IN STD_LOGIC;
     H,L:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
     SOUND1:OUT STD_LOGIC);

```

```

END DJSQ;
ARCHITECTURE DJSQ_ARC OF DJSQ IS
BEGIN
PROCESS(CLK1,EN)
VARIABLE HH,LL:STD_LOGIC_VECTOR(3 DOWNT0 0);
BEGIN
IF CLR='0' THEN
HH:="0011";
LL:="0000";
ELSE
IF CLK1'EVENT AND CLK1='1'THEN
IF EN='0'THEN
IF LL=0 AND HH=0 THEN
SOUND1<='1';
ELSIF LL=0 THEN
LL:="1001";
HH:=HH-1;
ELSE
LL:=LL-1;
END IF;
ELSE
SOUND1<='0';
HH:="0011";
LL:="0000";
END IF;
END IF;
END IF;
H<=HH;
L<=LL;
END PROCESS;
END DJSQ_ARC;

```

Code 7:DISPLAY1

功能：译码模块第二段程序源代码

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DISPLAY1 IS
    PORT(H,L:IN STD_LOGIC_VECTOR(3 DOWNT0 0);
         BCD1,BCD2:OUT STD_LOGIC_VECTOR(6 DOWNT0 0));
    END DISPLAY1;
ARCHITECTURE B OF DISPLAY1 IS
    BEGIN

```

```

PROCESS(H)
BEGIN
    CASE H IS
        WHEN "0000"=>BCD1<="1000000";
        WHEN "0001"=>BCD1<="1111001";
        WHEN "0010"=>BCD1<="0100100";
        WHEN "0011"=>BCD1<="0110000";
        WHEN "0100"=>BCD1<="0011001";
        WHEN "0101"=>BCD1<="0010010";
        WHEN "0110"=>BCD1<="0000010";
        WHEN "0111"=>BCD1<="1111000";
        WHEN "1000"=>BCD1<="0000000";
        WHEN "1001"=>BCD1<="0010000";
    WHEN OTHERS => BCD1 <="1111111";
    END CASE;
END PROCESS;
PROCESS(L)
BEGIN
    CASE L IS
        WHEN "0000"=>BCD2<="1000000";
        WHEN "0001"=>BCD2<="1111001";
        WHEN "0010"=>BCD2<="0100100";
        WHEN "0011"=>BCD2<="0110000";
        WHEN "0100"=>BCD2<="0011001";
        WHEN "0101"=>BCD2<="0010010";
        WHEN "0110"=>BCD2<="0000010";
        WHEN "0111"=>BCD2<="1111000";
        WHEN "1000"=>BCD2<="0000000";
        WHEN "1001"=>BCD2<="0010000";
    WHEN OTHERS => BCD2 <="1111111";
    END CASE;
END PROCESS;
END B;

```

Code 8:FENPIN

功能：分频程序源代码

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY FENPIN IS
    PORT(CLK:IN STD_LOGIC;
          CLK1:OUT STD_LOGIC);
END FENPIN;

```

```

ARCHITECTURE clk_div_behavior OF FENPIN IS
    SIGNAL counter:STD_LOGIC_VECTOR(24 DOWNTO 0);
    SIGNAL temp:STD_LOGIC;
BEGIN
    PROCESS(CLK)
    BEGIN
        IF(CLK'EVENT AND CLK='1')THEN
            IF(counter="1011111010111100000111111")THEN
                counter<="000000000000000000000000";
                temp<=NOT temp;
            ELSE
                counter<=counter+1;
            END IF;
        END IF;
    END PROCESS;
    CLK1<=temp;
END clk_div_behavior;

```