



数字电子技术基础 实验报告

小组成员： 夏明坤 2021300020

小组成员： 辛一然 2021300072

组 号： 7

摘要

数字电子技术实验是对数字电子技术理论课的巩固和拓展延伸。实验课使得理论课上学到的知识在实践中得到检验，同时带给我们更多理论课所无法涵盖的知识和经验。每一次提前预习让我们得以从容应对老师设置的附加题；每一次遇到难题时，我们查阅资料、讨论思考，不断尝试新的解决方法；而当每一次实验成功的那一刻，都让我们感到无比的快乐。

本学期实验课一共包含七次实验，从简单到复杂，从基础到深入，带领我们清晰地领略整门课程的思维脉络：

(1) 新手上路——门电路逻辑变换

实验一的主要目的是通过制作全加器来学习 Quartus 软件的使用和 FPGA 的烧录。通过画真值表、写逻辑函数、转换成原理图，进行波形仿真和在开发板上验证，培养了我们工程文件管理的层次性和细致性的要求，为后续的复杂实验打下了良好的基础。

(2) 崭露头角——组合电路设计和时序电路设计

实验二的组合电路设计和实验三の時序电路设计是整门课程最关键的两个基石。通过中规模器件实现全加器和全减器，我们温习巩固了通过真值表写函数表达式、使用卡诺图化简、对变量进行降维的思想等知识的学习；而计数器、组合电路实现序列发生器的运用，以及利用移位寄存器实现彩灯花型变换，进一步加深我们了对时序过程的理解，巩固了理论课程中所学内容。

(3) 游刃有余——基于硬件描述语言电路设计和 FPGA 的 ROM 使用

在这两个实验中，我们学习并使用了新的表达工具——VHDL 语言和 ROM。引入 VHDL 语言后，我们可以通过模块化设计快速实现门电路的复杂连线过程；ROM 可以存储只读数据，在实验中通过计数模块的驱动进行显示。另外学习了宏函数和模块配置等 Quartus 的新功能，使我们能够更加游刃有余地进行实验设计。

(4) 小有名气——基于 FPGA 的信号发生器和模数转换电路

最后两个实验我们实现了 D/A 和 A/D 的转换，将数电实验与物理世界和现实世界之间架起沟通的桥梁。虽然这些实验并不难，但在实际操作中我们也遇到

了许多细节问题，如复杂且不稳定的连线、线长不一问题等。然而，我们通过不断努力我们查明原因并成功解决了这些问题。

通过这门课程，我们收获的不仅仅是实现数电实验要求的技术能力，更重要的是借此机会深刻感受到了**将设计转化为实际应用的乐趣**。数电实验的独特魅力深深激发了我们的好奇心和求知欲，培养我们团队协作精神，让我们沉浸在探索的快乐中，不断驱使着我们追求更广阔、更深层次的知识与技术天地。

关键词：数电实验；FPGA；电路设计与验证；课程脉络；理论与实践；

目录

摘要.....	2
目录.....	4
实验一 门电路逻辑变换.....	6
一、实验目的.....	6
二、实验要求.....	6
三、实验设备.....	6
四、实验原理.....	6
五、实验内容.....	8
六、实验过程中的问题.....	9
七、心得体会.....	9
实验二 组合电路设计.....	11
一、实验目的.....	11
二、实验要求.....	11
三、实验设备.....	11
四、实验原理.....	11
五、实验内容.....	14
六、实验过程中的问题.....	15
七、心得体会.....	16
实验三 基于原理图的时序电路设计.....	17
一、实验目的.....	17
二、实验要求.....	17
三、实验设备.....	17
四、实验原理.....	17
五、实验内容.....	18
六、实验过程中的问题.....	21
七、心得体会.....	22
实验四 基于硬件描述语言电路设计.....	23
一、实验目的.....	23
二、实验要求.....	23
三、实验设备.....	24
四、实验原理.....	24
五、实验内容.....	24
六、实验过程中的问题.....	31
七、心得体会.....	31
实验五 FPGA 的 ROM (IP 核) 使用.....	32
一、实验目的.....	32
二、实验要求.....	32
三、实验设备.....	32
四、实验原理.....	33

五、实验内容.....	33
六、实验过程中的问题.....	38
七、心得体会.....	38
实验六 基于 FPGA 的信号发生器.....	40
一、实验目的.....	40
二、实验要求.....	40
三、实验设备.....	40
四、实验原理.....	40
五、实验内容.....	42
六、实验过程中的问题.....	45
七、心得体会.....	45
实验七 基于 FPGA 的模数转换电路.....	46
一、实验目的.....	46
二、实验要求.....	46
三、实验设备.....	46
四、实验原理.....	47
五、实验内容.....	47
六、实验过程中的问题.....	50
七、心得体会.....	50
参考文献.....	51

实验一 门电路逻辑变换

一、实验目的

- 1.熟悉 QuartusII13.0 的使用方法；
- 2.熟悉 FPGA 的各接口与功能；
- 3.回顾与非门的逻辑功能，实现一位全加器。

二、实验要求

用门电路设计实现一位全加器，用 FPGA 实现电路测试逻辑功能。

三、实验设备

- 1.Quartus II 13.0 软件；
- 2.FPGA 开发板。

四、实验原理

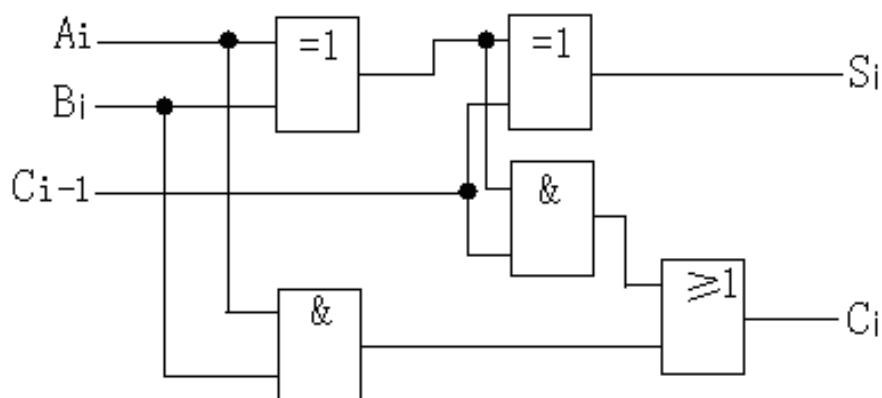
1.全加器

全加器是一种逻辑电路，用于执行二进制加法运算。它接收两个待相加的二进制位和一个进位信号作为输入，并输出当前位的和以及向更高位的进位信号。全加器可以用于构建多位的二进制加法器，实现更复杂的算术运算。

2.需要的逻辑函数

- 1) 逻辑与（AND）表示只有当所有条件都满足时，结果为真；
- 2) 逻辑非（NOT）表示取反，即真变为假，假变为真；
- 3) 逻辑与非（NAND）是指当所有输入都为真时结果为假，否则结果为真。

3.原理图



4.真值表

A	B	C_{i-1}	S	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

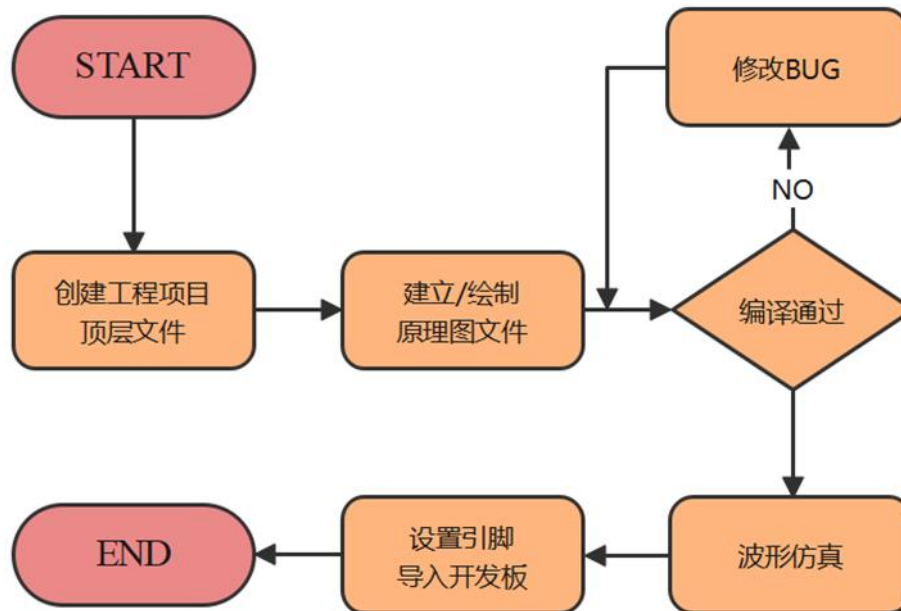
5.逻辑表达式

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$$

五、实验内容

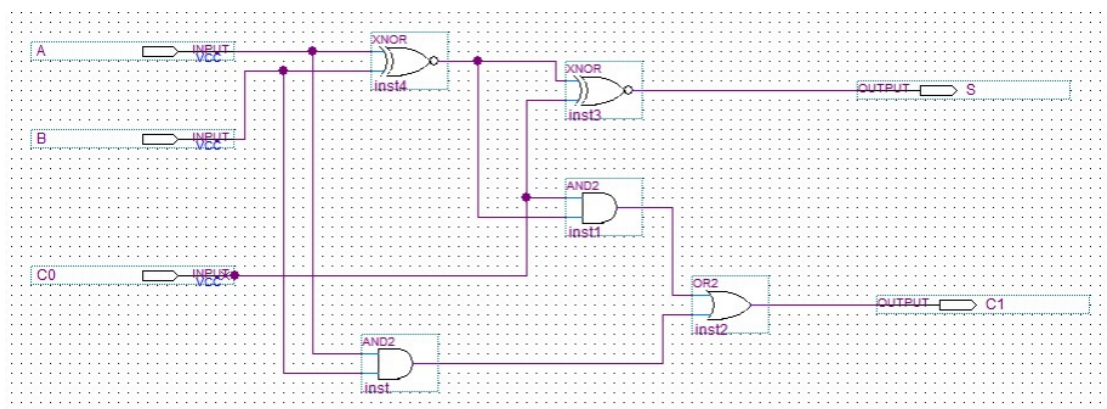
1.Quartus II 软件的使用



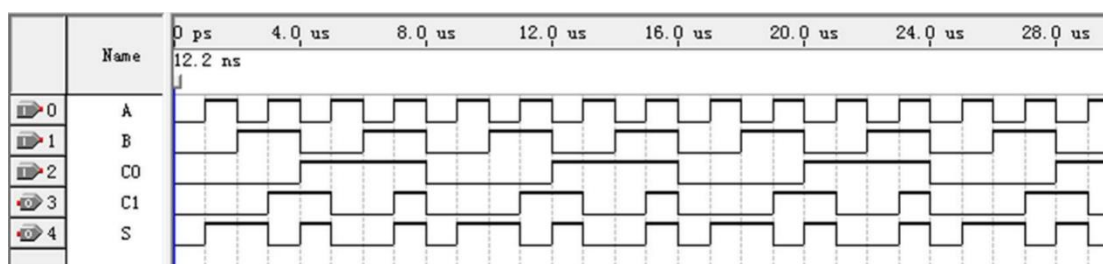
说明:

- 1) 工程文件的层次性。层次性的命名管理使得工程更加清晰、易于管理和维护;
- 2) 工程文件的细致性。在 Quartus II 中创建工程时, 需要考虑各种细节和参数设置, 以确保工程的正确性和性能。

2.电路图



3. 及波形仿真



4. 在实验开发板 DE0 上验证一位全加器的逻辑



输入 111, 输出 11; 输入 101, 输出 01; 输入 001, 输出 10; 输入 011, 输出 01。

六、实验过程中的问题

1. Quartus II 文件管理

本次实验原理上不难, 时间主要花在 Quartus 软件的使用上. 总的来说很多问题是文件命名不能用中文、首位不能是数字、器件命名不正确、重复;

2. Quartus II 版本兼容性

仿真需要 III 代, 烧录需要 IV 代, 如果没有跟换, 可能会导致无波形或无功能的问题;

3. Quartus II 无记忆性

对工程的任何更改, 包括原理图、波形图和绑定引脚, 都需要进行保存和编译, 否则看着有实际没有, 容易试验完导致文件丢失。

七、心得体会

1. 通过这个实验, 我初步了解了 FPGA 设计的过程和 Quartus II 工具的使用, 体验到了将设计转换为实际硬件的过程。相比于电基实验和模电实验, 我发现数

电实验有着独特的魅力，更吸引我。

2.通过这次实验，初次接触了如何用软件设计电路原理图，并通过原理图设计来操控开发板实现电路变换，从完全不懂到一步一步跟着教程做出实验结果，很有成就感。平时课本上的原理变成了生动可观、可设计可验证的软硬件操作，也非常有趣。

实验二 组合电路设计

一、实验目的

- 1.掌握数据选择器 74153 的逻辑功能及其基本应用，通过实验的方法学习数据选择器的电路结构和特点；
- 2.掌握三线八线译码器 74138 的使用方法。

二、实验要求

- 1.用 74138 三线八线译码器和 7420 与非门，用原理图输入方法实现一位全加器。Quartus II 实现波形仿真和下载开发板验证；
- 2.用 74153 双四数据选择器和 7400 与非门，用原理图输入方法实现一位全减器。Quartus II 实现波形仿真和下载开发板验证；
- 3.在要求 1 和要求 2 的基础上，自选门电路或组合逻辑电路，用 4 个开关作为控制端，当控制开关为一名组员学号的最后一位时实现一位全加器；当控制开关为另一名组员学号的最后一位时实现一位全减器。（如学号分别是 2021301809 和 2021301804 时，控制开关输入 9 实现全加器，控制开关输入 4 实现全减器灯，其他情况，输出为 0）。

三、实验设备

- 1.QuartusII13.0 软件；
- 2.FPGA 开发板。

四、实验原理

1.全加器

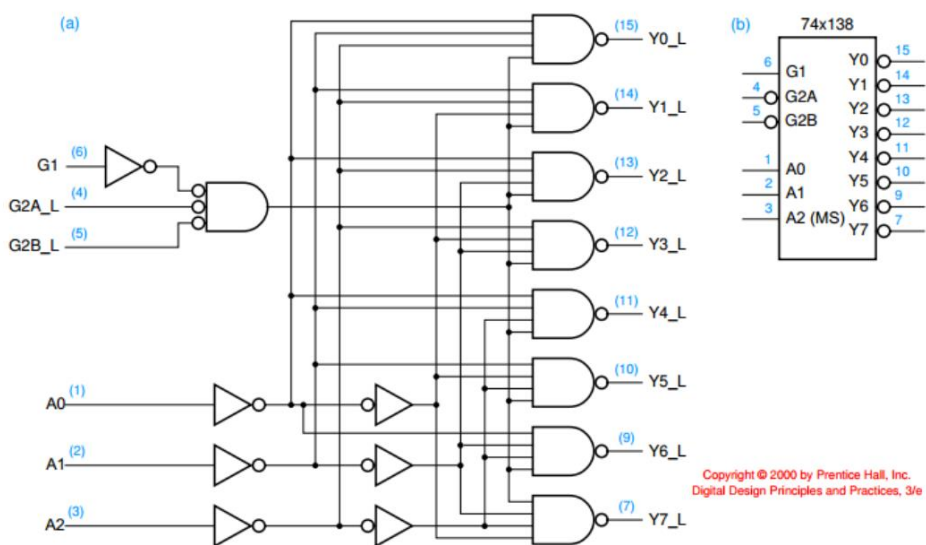
见上一个实验。

2.全减器

全减器是一种逻辑电路，用于计算二进制数的减法操作。它接受两个输入，被减数和减数，并产生两个输出，差值和借位。

3.使用器件

(1) 74138 三线八线译码器

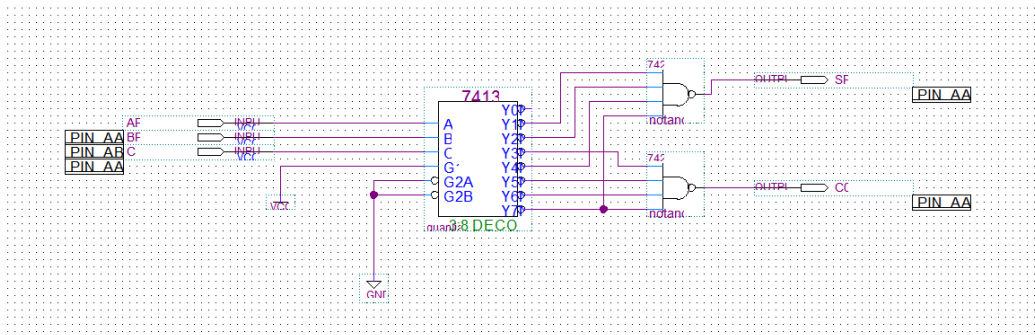


74138 三线八线译码器的三输入正好满足全加器三输入的要求，输出需要增加额外的门电路来满足，推导过程如下：

$$S = \sum m(1, 2, 4, 7) = \overline{Y_1} + \overline{Y_2} + \overline{Y_4} + \overline{Y_7} = \overline{Y_1 Y_2 Y_4 Y_7}$$

$$C_i = \sum m(3, 5, 6, 7) = \overline{Y_3} + \overline{Y_5} + \overline{Y_6} + \overline{Y_7} = \overline{Y_3 Y_5 Y_6 Y_7}$$

原理图如下：



(2) 74153 双四数据选择器

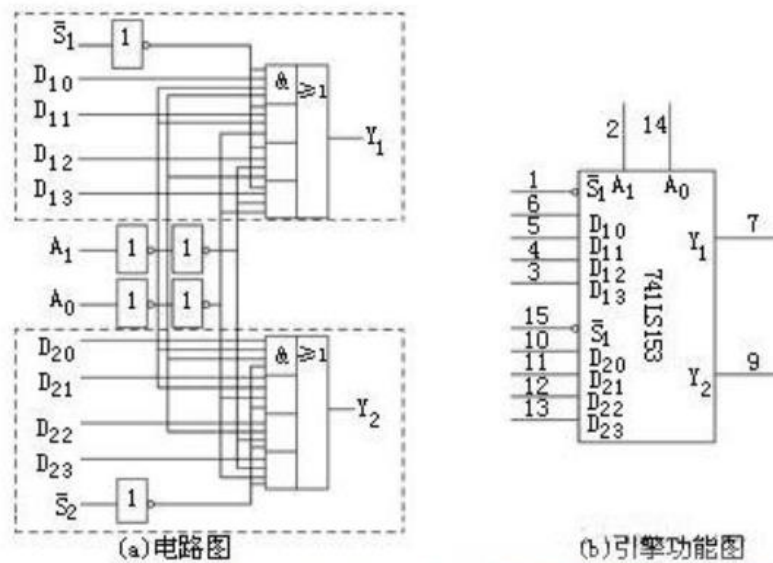


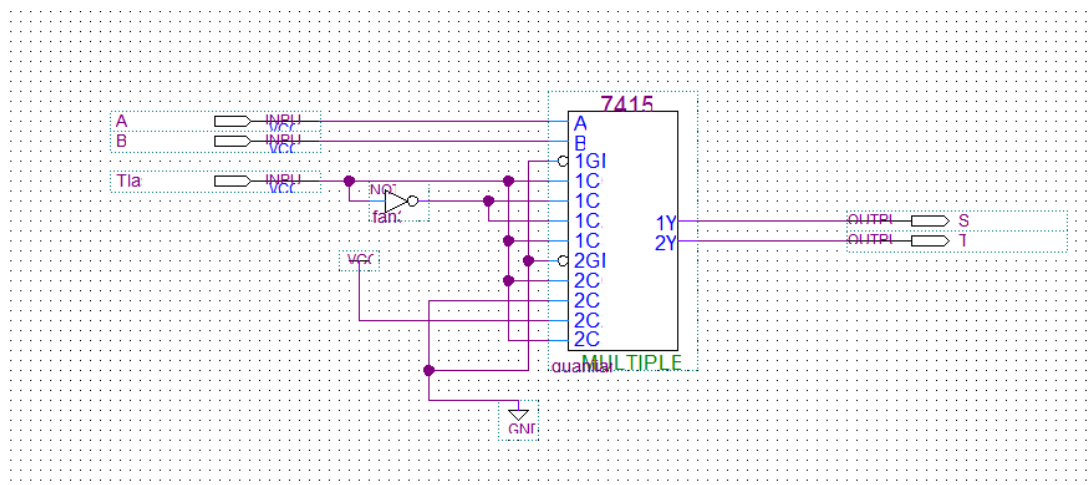
图3 74LS153双4选1数据选择器

使用 74153 双四数据选择器来实现来实现全减器，需要采用“降维”思想，将 D 端口和 C_i 融合一起考虑。

真值表如下：

A (被减数)	B (减数)	C_i	Y	降维 D_{1i}	C_o	降维 D_{2i}
0	0	0	0	C_i	0	C_i
0	0	1	1		1	
0	1	0	1	$\overline{C_i}$	1	1
0	1	1	0		1	
1	0	0	1	$\overline{C_i}$	0	0
1	0	1	0		0	
1	1	0	0	C_i	0	C_i
1	1	1	1		1	

电路图如下：



五、实验内容

注：由于要求3是要求1和2的综合并且要求1和2的电路在上面，下面将详细描述要求3。

要求三：设置一个控制开关，控制全加器和全减器的选择工作。

1.控制开关

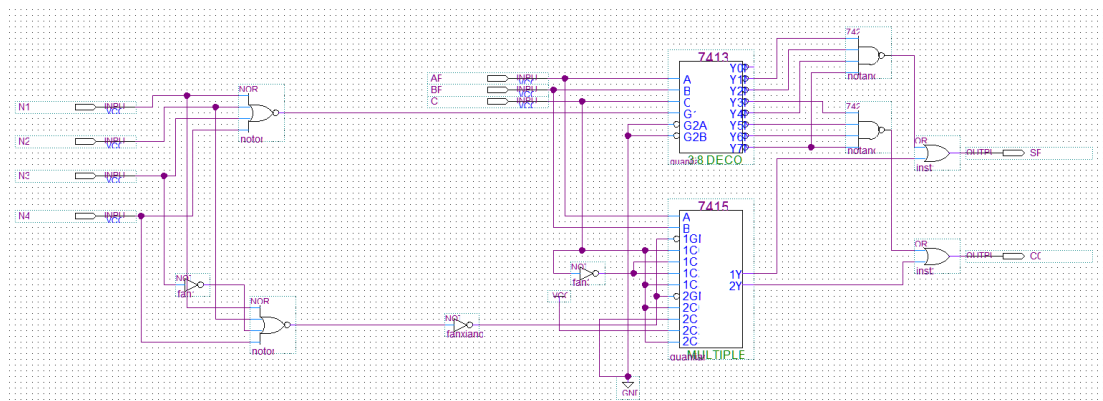
学号末尾0（0000）控制全加器，2（0010）控制全减器

$$G_+ = \overline{N_1} + \overline{N_2} + \overline{N_3} + \overline{N_4} = \overline{N_1 + N_2 + N_3 + N_4}$$

$$G_- = N_1 + N_2 + \overline{N_3} + \overline{N_4} = \overline{\overline{N_1 + N_2 + N_3 + N_4}}$$

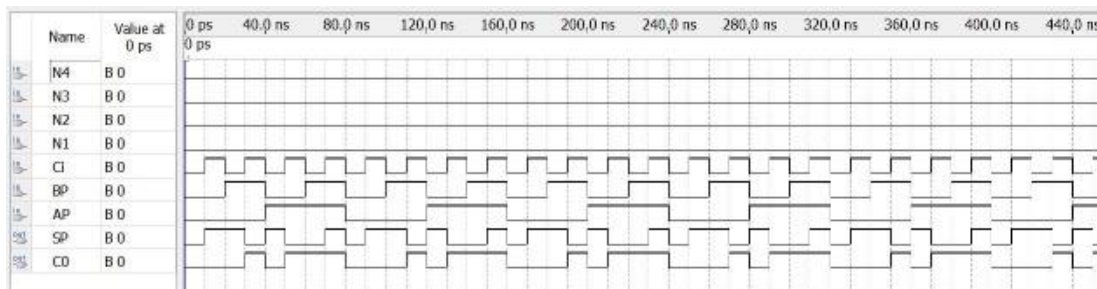
其中N1、N2、N3、N4依次从高位到低位。

2.原理图

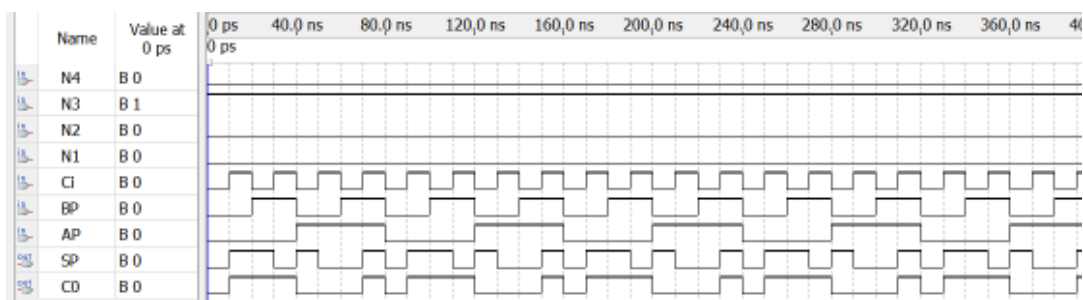


3.波形仿真

下图为全加器



下图为全减器



4.在实验开发板 DE0 上验证

第一层为 0000 控制开关的全加器，
输入 111，输出 11，输入 110，输出 01；

第二行为 0010 控制开关的全减器，
输入 110，输出 00，输入 111，输出 11；

第三行为 1111 控制开关的全加器/全
减器，输入 000，输出 00，输入 101，输出 00，满足要求——其他情况，输出为 0。



六、实验过程中的问题

- 1.第一次将课本所学写入变成现实，总会有些合理但很复杂的地方，比如开关 G-（减法器）的设置就复杂了，可以省去那个单独的反相器；
- 2.由于这次门电路使用较多，命名 inst 容易重复，多次修改后成功编译；
- 3.在连线时不能搭接，虽然是连上，符合逻辑，但 Quartus 有可能会报错，

要关注细节。

七、心得体会

在进行画图编译之前，务必仔细分析和理解问题，只有了解了输入和输出之间的逻辑关系，确定所需的逻辑运算和功能，才能正确编译。另外要善于总结组合电路设计技巧，“降维”思想的应用让全减器的设计变得很简单

实验三 基于原理图的时序电路设计

一、实验目的

- 1.通过实验的方法学习时序电路的基本设计方法，掌握时序电路的功能特点；
- 2.学会使用 Quartus II 软件内嵌宏函数 lpm_counter 实现分频功能。

二、实验要求

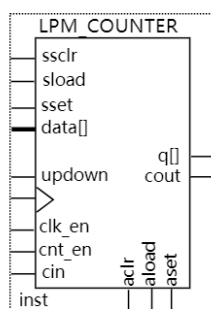
- 1.参照参考内容,用 Quartus II 软件内嵌宏函数 lpm_counter 实现 50M 分频,输出频率为 1Hz 秒脉冲信号,用实验板上绿色 LED 灯观察;
- 2.参照参考内容中数码管显示控制电路设计方法,用计数器、七段译码器和若干门电路,用原理图输入方法实现在一个 7 段数码管上显示序列:2021300020;
- 3.用二进制计数器、移位寄存器和若干门电路,用原理图输入方法实现彩灯控制器电路设计;
- 4.将要求 1 和 2 同时在实验电路上实现。

三、实验设备

- 1.QuartusII13.0 软件;
- 2.FPGA 开发板。

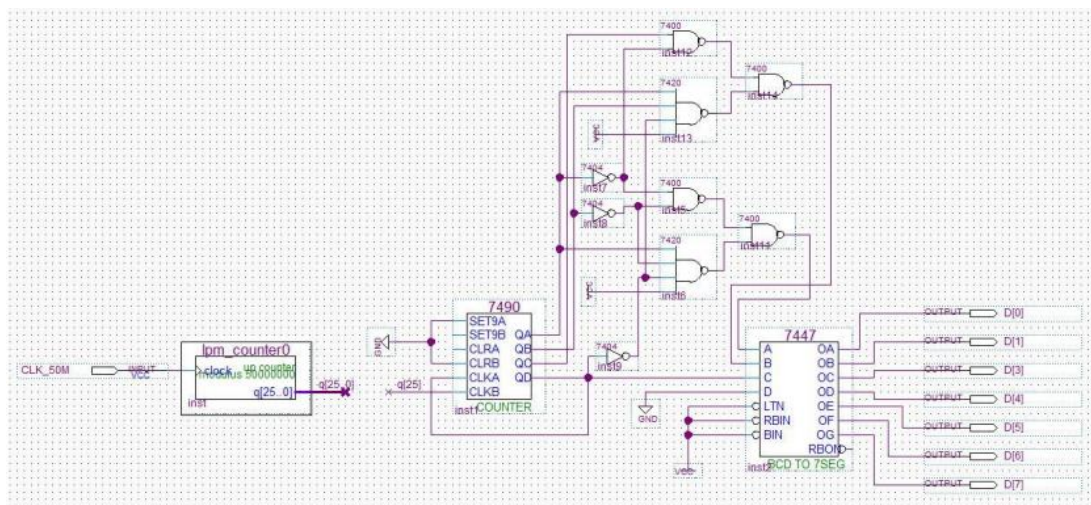
四、实验原理

1.lpm_counter



lpm_counter 是一种内嵌宏函数,用于实现计数器功能,允许设计者快速创建和配置计数器,以满足各种应用需求。在 Quartus II 软件中,可以通过添加和配置 lpm_counter 实例来创建多个计数器,并在设计中集成它们以满足特定的设计需求。

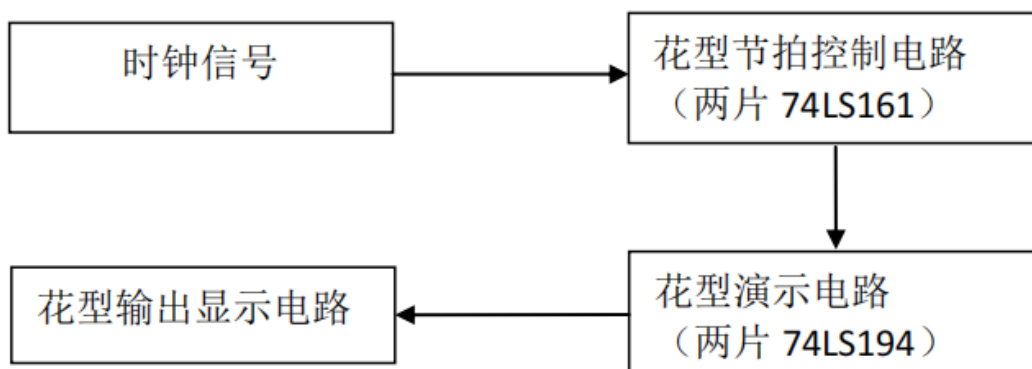
2.七段数码管控制电路



原理：lpm_counter 实现 50M 分频，取 q[25]作为 74LS290 的时钟信号输入，通过门电路实现逻辑函数的转化，通过 74LS47 译码器驱动共阳极数码管显示。

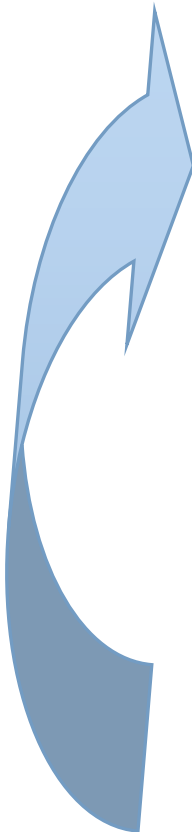
3.彩灯控制器

总体电路分为四大模块：模块一由 Quartus 宏函数 lpm_counter 提供时钟脉冲信号；模块二花型节拍控制电路由两片 74LS161 组成一个 32 进制计数器；模块三花型演示电路由两片 74LS194 来控制花型；模块四花型输出显示电路。总体原理框图如下：



五、实验内容

1.学号序列发生器



D	C	B	A	QD	QC	QB	QA	Number
0	0	0	0	0	0	1	0	2
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	1	0	2
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	1	3
0	1	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	1	0	2
1	0	0	1	0	0	0	0	0
1	0	1	0					

本实验我们采用异步置零法，则逻辑函数为：

$$Q_D = Q_C = 0$$

$$Q_B = \overline{D}\overline{C}\overline{B}\overline{A} + \overline{D}\overline{C}B\overline{A} + \overline{D}C\overline{B}\overline{A} + \overline{D}CB\overline{A} = \overline{B}\overline{D} + \overline{C}\overline{D}$$

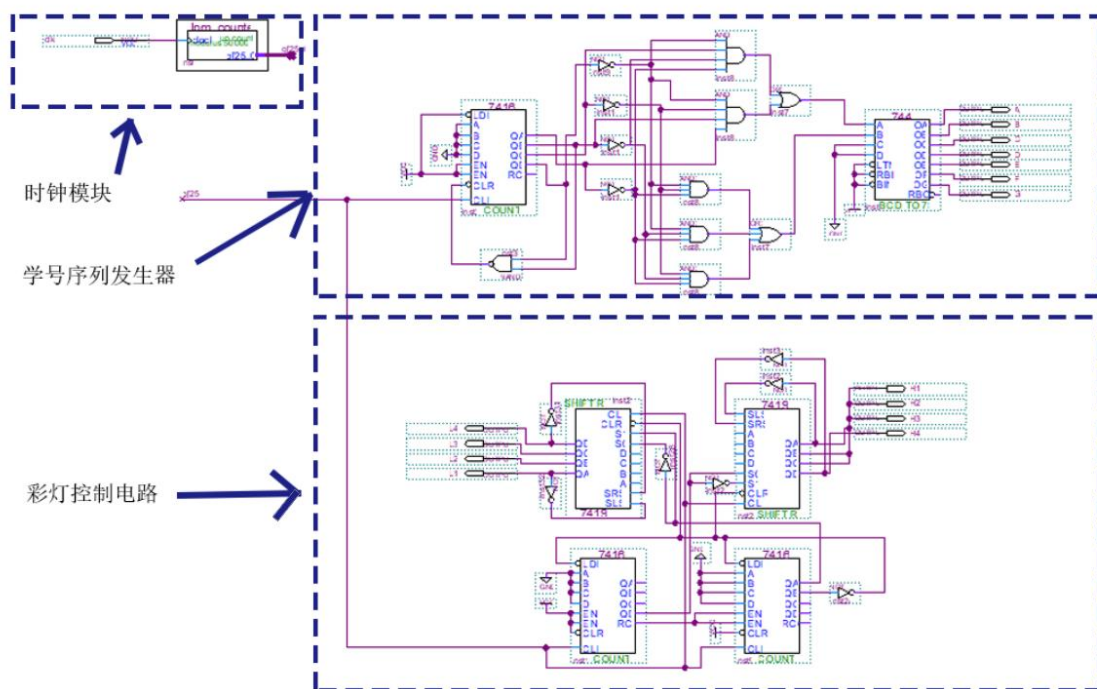
$$Q_A = \overline{D}\overline{C}\overline{B}A + \overline{D}\overline{C}B\overline{A} = \overline{B}\overline{C}\overline{D} + B\overline{C}\overline{D}$$

$Q_B Q_A$ 需要无关项+卡诺图化简，如下：

AB \ CD	00	01	11	10
00	1	1	x	1
01	0	0	x	0
11	0	0	x	x
10	1	0	x	x

AB \ CD	00	01	11	10
00	0	1	x	0
01	0	0	x	0
11	1	0	x	x
10	0	0	x	x

2.原理图



(1) 左上部分为按照步骤搭建的 50MHz-1Hz 的时钟 lpm_counter;

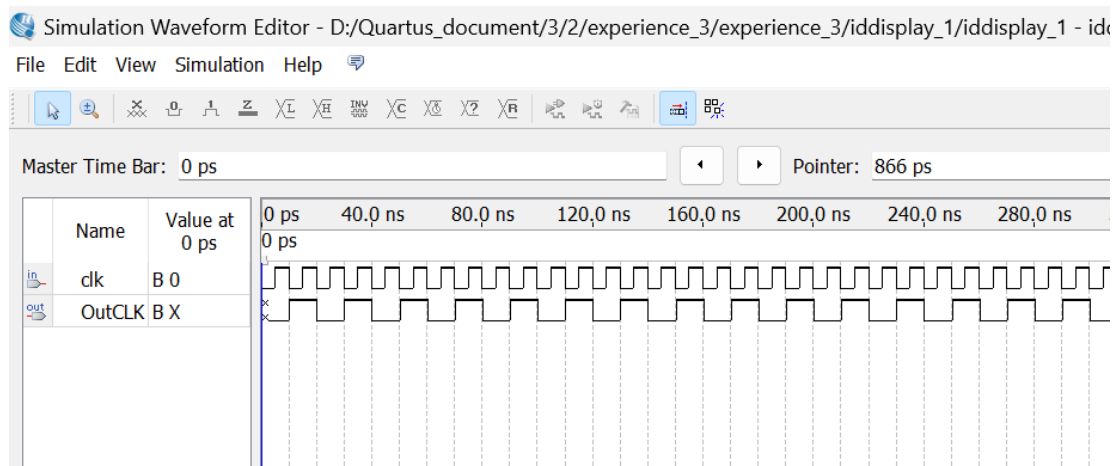
(2) 上半部分为学号序列 (2021300020) 发生器, 采用计数器+门电路形式完成, 也可以将门电路换成数据选择器、译码器或者后面的 VHDL 硬件编程语言完成;

(3) 下半部分为彩灯控制电路, 采用两个译码器和两个移位寄存器完成;

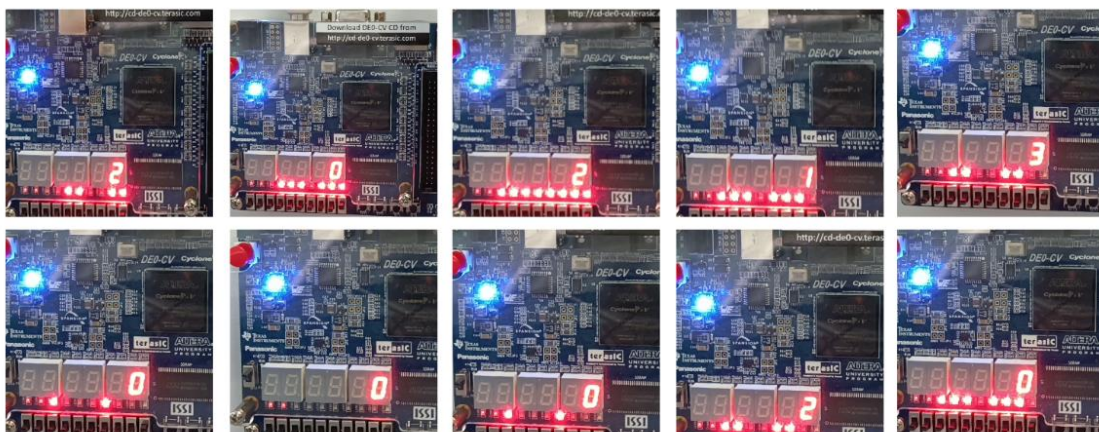
74194 的控制端 $S1=0, S0=1$ 时, 进行右移, $S1=1, S0=0$ 时, 进行左移, 通过 SR、SL 变化实现一定规律的花型;

74161 预置数 0000 (接地), 从 0000 开始计数, 两片计数器结合的模式为 32, 满足实验要求。

3.波形图 (显示分频效果+彩灯控制)



4.在实验开发板 DE0 上验证



如上所示，2021300020 学号序列正常产生，间接说明 lpm 正常工作。

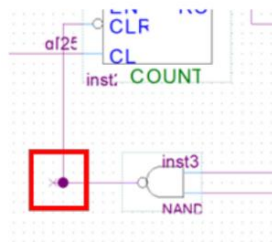
彩灯的变化过程：

花型I——由两边向中间对称性依次亮，全亮后仍由两边向中间依次灭；花型II——8 路灯分两半，从左自右顺次亮，再顺次灭；花型 III——8 路灯分两半，从右向左顺次亮，再从右向左顺次灭；花型 IV——由中间向两边对称性一次亮，全亮后仍由中间向两边依次灭，这四种花型循环出现。

六、实验过程中的问题

1.第一次做时，数码管显示的数字不符合预期，最后发现是没有分清逻辑变量的高位和低位，接反了。

2.门电路的方法过于复杂，容易连错还耗时间，不如后面学到的 VHDL 语言。3.经常会出现给一个元器件重复赋值的情况，后来发现是如图的情况。连线不够仔细，存在边角小瑕疵。



12014Net "gdfx_temp0", which fans out to "inst91", cannot be assigned more than one value
12014Net "gdfx_temp1", which fans out to "inst85", cannot be assigned more than one value
Quartus II 64-Bit Analysis & Synthesis was unsuccessful. 7 errors, 0 warnings

七、心得体会

1.这次实验让我更好理解时序的意义，在时序电路设计中，组件之间的时序关系特别绕。LPM_Counter 的计数操作可能会受到时钟信号的控制，而移位寄存器的移位操作可能会在特定的时钟边沿触发，思考起来就很复杂。

2.门电路真的很复杂，也容易接错，VHDL 语言学完之后就特别简单。

实验四 基于硬件描述语言电路设计

一、实验目的

- 1.学习硬件描述语言 VHDL 的使用方法,并通过原理图模块和顶层文件的结合实现特定功能;
- 2.熟悉分频电路的功能特点。

二、实验要求

1.学习并掌握硬件描述语言 VHDL;熟悉门电路的逻辑功能,并用硬件描述语言实现门电路的设计。参考“参考内容 1”中给出的与门源程序,编写一个异或门逻辑电路。1)用 Quartus II 波形仿真验证;2)下载到 DE0 开发板验证;

2.熟悉中规模器件译码器的逻辑功能,用硬件描述语言实现其设计。参考“参考内容 2”中给出的将 8421BCD 码转换成 0-9 的七段码译码器源程序,编写一个将二进制码转换成 0-E 的七段码译码器。1)用 Quartus II 波形仿真验证;2)下载到 DE0 开发板,利用开发板上的数码管验证;

3.熟悉时序电路计数器的逻辑功能,用硬件描述语言实现其设计。参考“参考内容 3”中给出的四位二进制计数器的源程序,编写一个计数器实现 0-E 计数。用 Quartus II 波形仿真验证;

4.熟悉分频电路的逻辑功能,并用硬件描述语言实现其设计。参考“参考内容 4”中给出的 50M 分频器的源程序,编写一个能实现占空比 50%的 5M50M 分频器即两个输出,输出信号频率分别为 10Hz 和 1Hz。下载到 DE0 开发板验证。

(提示:利用 DE0 板上已有的 50M 晶振作为输入信号,通过开发板上两个的 LED 灯观察输出信号)。电路框图如下:



5.利用已经实现的 VHDL 模块文件,顶层文件采用原理图设计方法,实现 abcdE+00072+Edcba 计数自动循环显示,频率 1Hz 和 10Hz 可以切换。(提示:

如何将 VHDL 模块文件在顶层原理图文件中引用，参考参考内容 5)；

6.让 FPGA 上的数码管循环显示 Edcba+00072，每个循环结束后暂停；然后拨动一开关，继续循环显示，然后循环停止拨动开关，重复。

三、实验设备

- 1.Quartus II 13.0 软件；
- 2.FPGA 开发板。

四、实验原理

VHDL (VHSIC Hardware Description Language) 是一种硬件描述语言，用于描述数字电路的结构和行为。VHDL 最初由美国国防部高级研究计划局 (VHSIC) 于 20 世纪 80 年代开发，旨在满足对复杂集成电路 (VLSI) 设计的需求。

VHDL 的语法结构类似于 C 语言和 LINGO 语言，基本结构由五个主要部分组成，如下图所示。我们主要会用到的是 2、3、4 三个部分。



五、实验内容

1.异或门

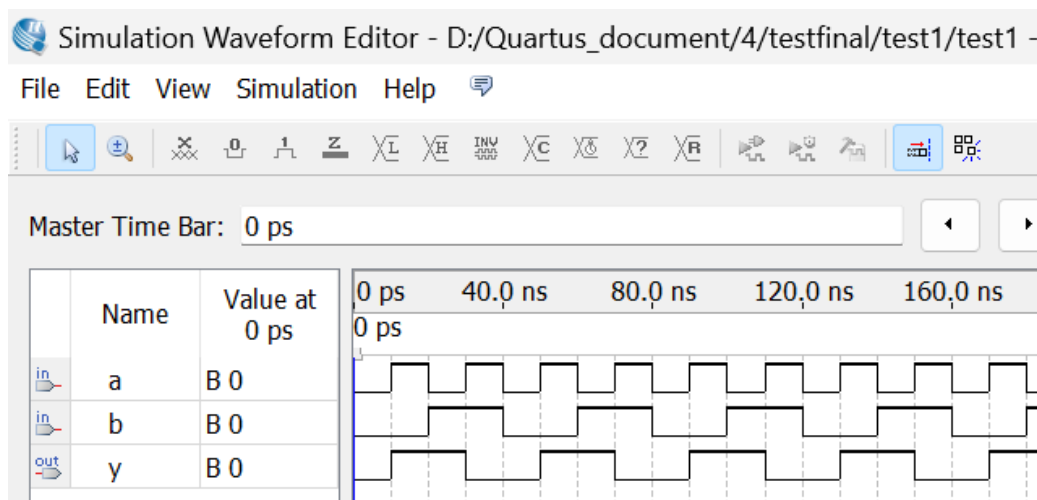
(1) VHDL 代码

```

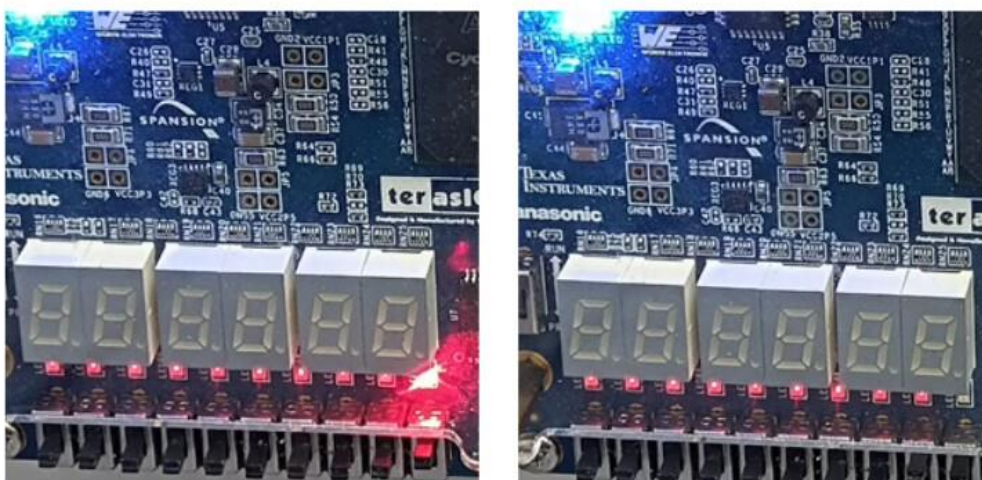
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity test1 is
5  port (
6      a : in std_logic;
7      b : in std_logic;
8      y : out std_logic
9  );
10 end test1;
11
12 architecture behavior of test1 is
13 begin
14     y <= a xor b;
15 end behavior;

```

(2) 波形验证



(3) 在实验开发板 DE0 上验证



左边开关状态相异，灯亮；右边开关状态相同，灯灭。

2.将二进制码转换成 0-E 的七段码译码器

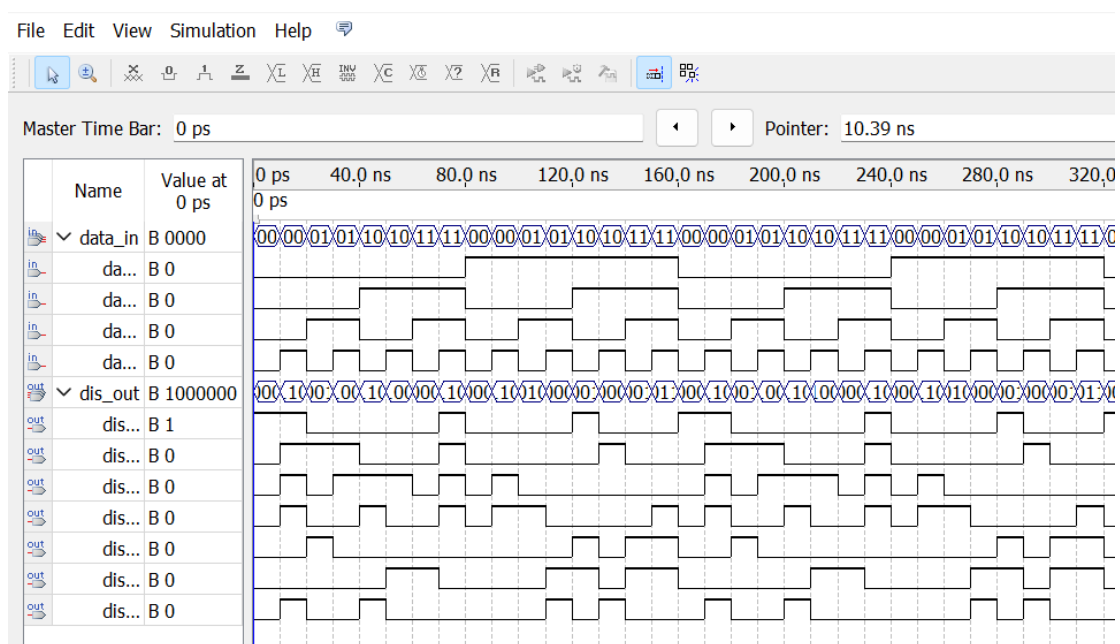
(1) VHDL 代码

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  ENTITY test2 IS
4  PORT (data_in:IN STD_LOGIC_VECTOR(3 DOWNTO 0);dis_out:OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
5  END test2;
6  ARCHITECTURE fwm OF test2 IS
7  BEGIN
8  PROCESS(data_in)
9  BEGIN
10 CASE data_in IS
11 when "0000" => dis_out <= "1000000"; -- 0
12 when "0001" => dis_out <= "1111001"; -- 1
13 when "0010" => dis_out <= "0100100"; -- 2
14 when "0011" => dis_out <= "0110000"; -- 3
15 when "0100" => dis_out <= "0011001"; -- 4
16 when "0101" => dis_out <= "0010010"; -- 5
17 when "0110" => dis_out <= "0000010"; -- 6
18 when "0111" => dis_out <= "1111000"; -- 7
19 when "1000" => dis_out <= "0000000"; -- 8
20 when "1001" => dis_out <= "0011000"; -- 9
21 when "1010" => dis_out <= "0001000"; -- A 10
22 when "1011" => dis_out <= "0000011"; -- B 11
23 when "1100" => dis_out <= "1000110"; -- C 12
24 when "1101" => dis_out <= "0100001"; -- D 13
25 when "1110" => dis_out <= "0000110"; -- E 14
26 when "1111" => dis_out <= "0001110"; -- F 15
27 when others => dis_out <= "1111111"; -- 灭灯, 不显示
28 --WHEN"0000" =>dis_out<="0001000"; --A
29 --WHEN"0001" =>dis_out<="0000011"; --B
30 --WHEN"0010" =>dis_out<="1000110"; --C
31 --WHEN"0011" =>dis_out<="0100001"; --D
32 --WHEN"0100" =>dis_out<="0000110"; --E
33 --WHEN"0101" =>dis_out<="0100100"; --2
34 --WHEN"0110" =>dis_out<="1000000"; --0
35 --WHEN"0111" =>dis_out<="0100100"; --2
36 --WHEN"1000" =>dis_out<="1111001"; --1
37 --WHEN"1001" =>dis_out<="0110000"; --3
38 --WHEN"1010" =>dis_out<="0000110"; --E
39 --WHEN"1011" =>dis_out<="0100001"; --D
40 --WHEN"1100" =>dis_out<="1000110"; --C
41 --WHEN"1101" =>dis_out<="0000011"; --B
42 --WHEN"1110" =>dis_out<="0001000"; --A
43 --WHEN OTHERS=>dis_out<="1111111";
44 END CASE;
45 END PROCESS;
46 END fwm;

```

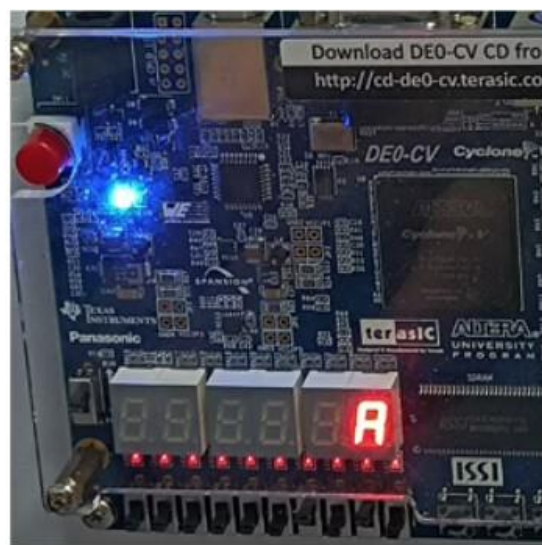
(2) 波形图仿真



(3) 在实验开发板 DE0 上验证



0100 表示 4;



1010 表示 A

3.四位二进制计数器

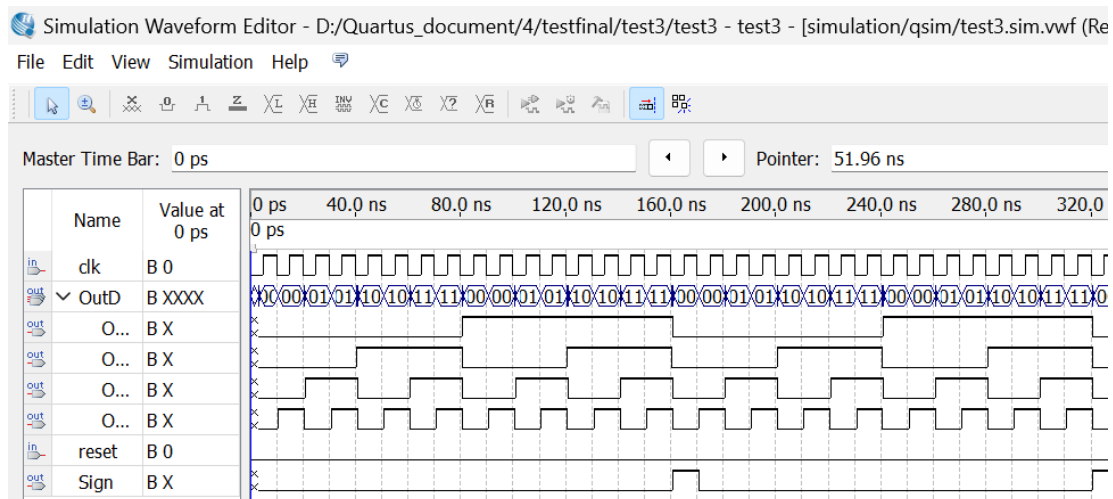
(1) VHDL 语言

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity test3 is --本实验为同步二进制加法计数器，模值为16
6  port (
7      clk :in std_logic; --时钟信号
8      reset :in std_logic; --复位信号
9      OutD :out std_logic_vector(3 downto 0); --0-15 4位宽的向量
10     Sign :out std_logic --提示新的循环
11 );
12 end test3;
13
14 architecture Behavior of test3 is
15     signal temp: std_logic_vector(3 downto 0); --temp表示一个4位宽的向量
16 begin
17     process (clk,reset)
18     begin
19         if (reset='1') then
20             temp<=(others => '0'); --temp <= '0000'
21             Sign<='0';
22         elsif (rising_edge(clk) and clk = '1') then --上升沿（从低电平到高电平的变化）
23             --elsif (falling_edge(clk) and clk = '1') then --下降沿（从高电平到低电平的变化）
24                 temp<=temp+1;
25                 Sign<='0';
26                 if (temp >= "1111") then
27                     temp<=(others => '0');
28                     Sign<='1';
29                 end if;
30             end if;
31         end process;
32         OutD<=temp;
33     end Behavior;
34

```

(2) 波形仿真



4.分频电路

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity test4 is
4  port (
5      clk : in std_logic;
6      enable : in std_logic;
7      out_10Hz : out std_logic;
8      out_1Hz : out std_logic
9  );
10 end test4;
11 architecture Behavior of test4 is
12     constant CLK_5M : natural := 2500000;
13     constant CLK_50M : natural := 25000000;
14
15     signal count_10Hz : natural range 0 to CLK_5M-1;
16     signal count_1Hz : natural range 0 to CLK_50M-1;
17
18     signal flipflop_10Hz : std_logic; --flipflop翻转器
19     signal flipflop_1Hz : std_logic;
20 begin
21     process (enable, clk)
22     begin
23         if enable = '1' then
24             if rising_edge(clk) then
25                 -- 10Hz分频
26                 if count_10Hz = CLK_5M - 1 then
27                     count_10Hz <= 0;
28                     flipflop_10Hz <= not flipflop_10Hz;
29                 else
30                     count_10Hz <= count_10Hz + 1;
31                 end if;
32
33                 -- 1Hz分频
34                 if count_1Hz = CLK_50M - 1 then
35                     count_1Hz <= 0;
36                     flipflop_1Hz <= not flipflop_1Hz;
37                 else
38                     count_1Hz <= count_1Hz + 1;
39                 end if;
40             end if;
41         else
42             flipflop_10Hz <= '0';
43             flipflop_1Hz <= '0';
44         end if;
45     end process;
46
47     --分配变量
48     out_10Hz <= flipflop_10Hz;
49     out_1Hz <= flipflop_1Hz;
50 end architecture Behavior;
51

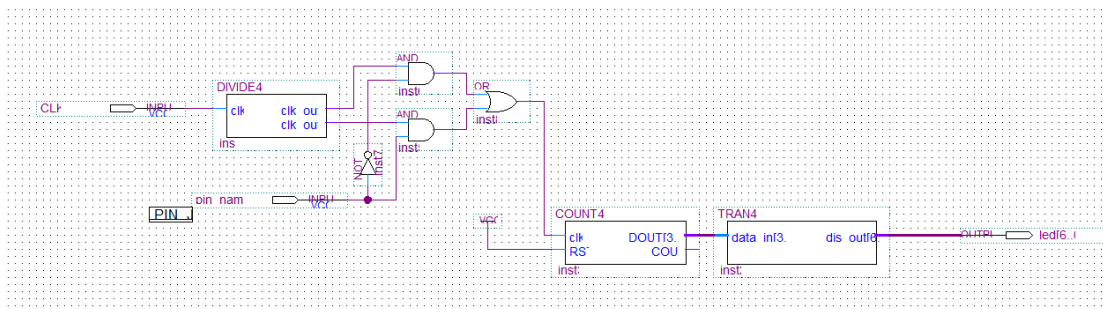
```

(2) 波形仿真

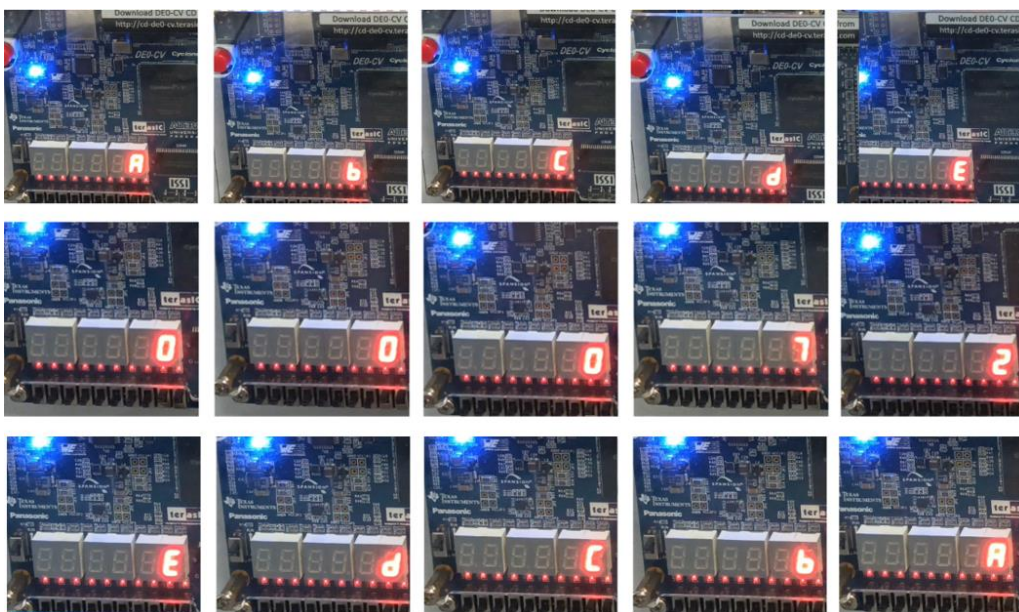
由于分频值过大，Quartus 波形图无法体现。

5.abcdE+00072+Edcba 计数自动循环显示

(1) 顶层文件

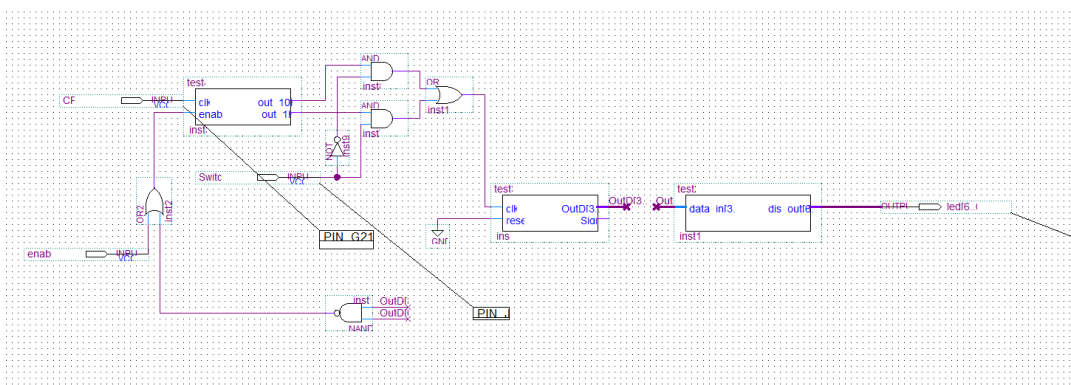


(2) 在实验开发板 DE0 上验证



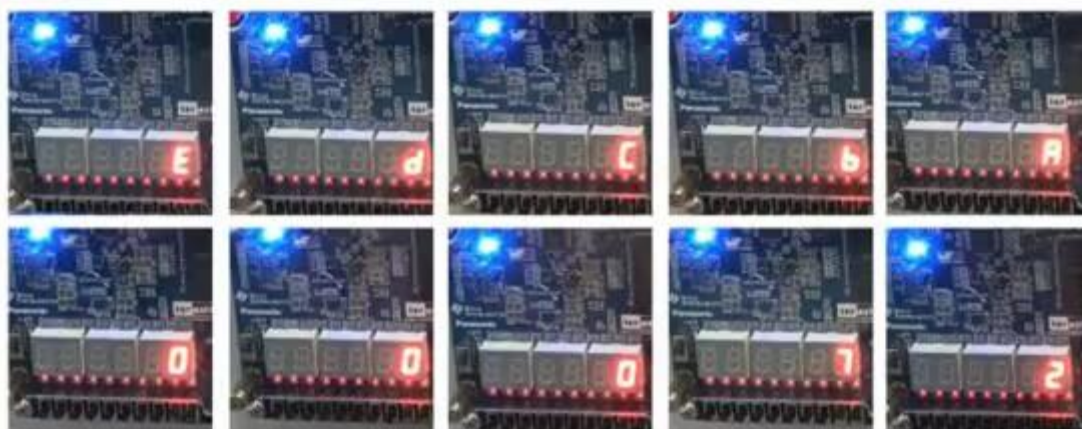
6.Edcba+00072 计数自动循环显示+循环结束拨动开关

(1) 顶层文件



在要求 5 基础上加了新的功能开关，通过真值表可以得到

(2) 在实验开发板 DE0 上验证



经当堂验收，所有设计符合要求。

六、实验过程中的问题

- 1.由于不熟悉 VHDL 的语法，刚开始频频报错，后来通过查阅大量资料和自己数模编程的经验得以解决；
- 2.有时候代码没错总是报错，要么重新打开，要么自己打一遍代码，就编译成功了，可能 Quartus 内部有 bug，Reset 就好了；
- 3.VHDL 编写器件时，实体名，即 Entity 后的名字，要和把它保存为的文件名，即*.vhd 文件的名字一致。

七、心得体会

- 1.通过学习掌握硬件语言——VHDL，在 Quartus 软件上通过 VHDL 设计出所需要的具有特定功能的元器件，放在设计好的电路图中实现相应的功能。使用 VHDL 使得功能实现变得更加轻松，更加灵活，更加快捷！
- 2.通过这节课认识了一个新的硬件描述语言，学会更加高效地完成需要的电路设计。

实验五 FPGA 的 ROM (IP 核) 使用

一、实验目的

- 1.掌握只读存储器 ROM 的工作原理和使用方法；
- 2.了解一般 ROM 的连接方法，加深总线概念的理解；
- 3.熟悉 MIF 文件的生成方法。

二、实验要求

1.配置宽度为 8 位的 ROM，并在 ROM 中存储 256 个地址的正弦波数，利用系统时钟作为计数器计数脉冲，利用计数器或硬件描述语言生成的计数模块的输出依次扫描 ROM 的地址端，将 ROM 中保存的数据按照 1Hz 的频率输出，通过 DE0 开发板上的 8 位 LED 灯查看结果并验证；

2. 配置存储学号（F+00020+E+00072+d）的 ROM，利用计数器的输出依次扫描 ROM 的地址端，将 ROM 中保存的数据按照 1Hz 的频率输出，通过 DE0 开发板上的 1 个七段数码管查看结果并验证；

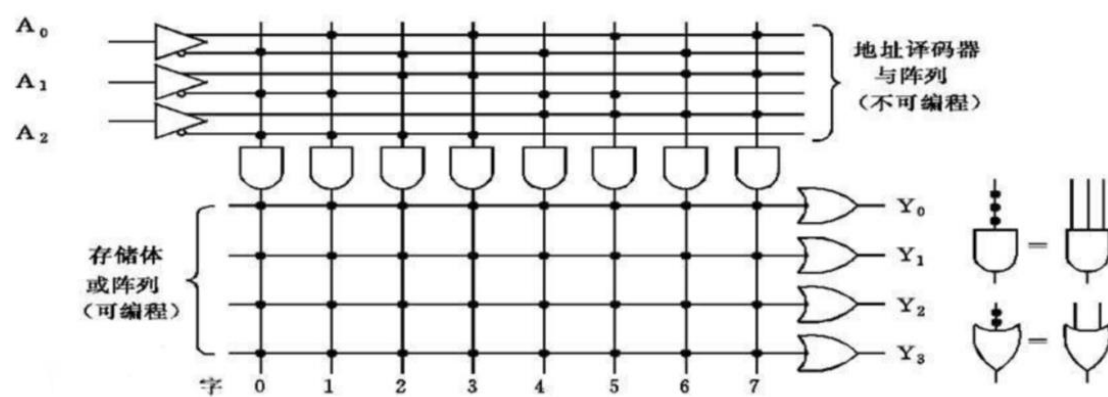
3. 将要求 1 中的数值转换成三个七段数码管，分百位、十位、个位将正弦波值输出。

三、实验设备

- 1.Quartus II 13.0 软件；
- 2.FPGA 开发板。

四、实验原理

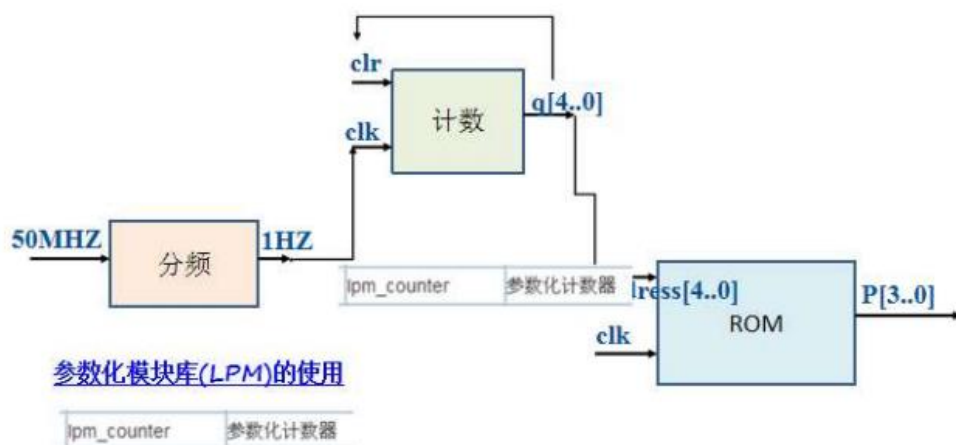
ROM 由三部分组成：地址译码器、存储矩阵和读出选择电路。当地址译码器选中某个字节后该字节的若干位被同时读出。以下是 ROM 的点阵图表示：



五、实验内容

1. 实验设计

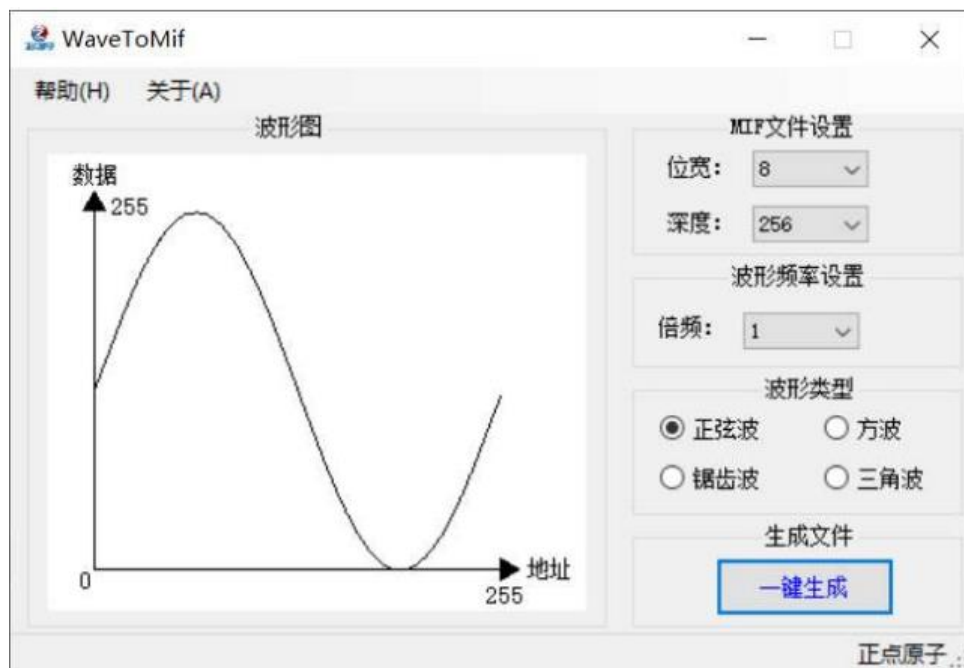
本实验通过分频器产生的时钟信号驱动计数器工作，计数模块扫描 ROM 中的数据，原理框图如下：



2. 模块配置

(1) MIF 文件生成

这是 Wave to Mif 软件的配置：



(2) ROM 的配置

按照参考内容生成 ROM 的操作：

Tools->MegaWizard Plug-in Manager->Memory Compiler->ROM:1 PORT

(3) 计数模块

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  ENTITY ROMuse_count IS
6  PORT (clk, RST: IN STD_LOGIC;
7        DOUT: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
8        COUT: OUT STD_LOGIC);
9  END ROMuse_count;
10
11 ARCHITECTURE fwm OF ROMuse_count IS
12   SIGNAL Q1: STD_LOGIC_VECTOR(7 DOWNTO 0);
13 BEGIN
14   PROCESS (clk, RST)
15   BEGIN
16     IF RST='0' THEN Q1<=(OTHERS=>'0'); COUT<='0';
17     ELIF clk' EVENT AND clk='1' THEN
18       Q1<=Q1+1;
19       COUT<='0';
20       IF Q1>="11111111" THEN Q1<=(OTHERS=>'0'); COUT<='1';
21     END IF;
22   END IF;
23 END PROCESS;
24 DOUT<=Q1;
25 END fwm;

```

(4) 分频模块

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY ROMuse_divide IS
5  PORT (clk:IN STD_LOGIC;
6        clk_out1:OUT STD_LOGIC;clk_out2:OUT STD_LOGIC);
7  END ROMuse_divide;
8
9  ARCHITECTURE fwm OF ROMuse_divide IS
10  CONSTANT m1:INTEGER:=2500000;
11  CONSTANT m2:INTEGER:=25000000;
12  SIGNAL tmp1:STD_LOGIC;
13  SIGNAL tmp2:STD_LOGIC;
14  BEGIN
15  PROCESS (clk)
16  VARIABLE cout1:INTEGER:=0;
17  VARIABLE cout2:INTEGER:=0;
18  BEGIN
19  IF clk'EVENT AND clk='1'THEN
20  cout1:=cout1+1;
21  IF cout1<=m1 THEN tmp1<='0';
22  ELSIF cout1<m1*2 THEN tmp1<='1';
23  ELSE cout1:=0;
24  END IF;
25  END IF;
26  IF clk'EVENT AND clk='1'THEN
27  cout2:=cout2+1;
28  IF cout2<=m2 THEN tmp2<='0';
29  ELSIF cout2<m2*2 THEN tmp2<='1';
30  ELSE cout2:=0;
31  END IF;
32  END IF;
33  END PROCESS;
34  clk_out1<=tmp1;
35  clk_out2<=tmp2;
36  END fwm;

```

(5) 7 段数码管译码器 VHDL 语言

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY TRAN1 IS
5  | PORT (data_in:IN STD_LOGIC_VECTOR(3 DOWNTO 0);dis_out:OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
6  | END TRAN1;
7
8  ARCHITECTURE fwm OF TRAN1 IS
9  BEGIN
10 | PROCESS(data_in)
11 | BEGIN
12 | CASE data_in IS
13 | WHEN"0000" =>dis_out<="1000000"; --0
14 | WHEN"0001" =>dis_out<="1111001"; --1
15 | WHEN"0010" =>dis_out<="0100100"; --2
16 | WHEN"0011" =>dis_out<="0110000"; --3
17 | WHEN"0100" =>dis_out<="0011001"; --4
18 | WHEN"0101" =>dis_out<="0010010"; --5
19 | WHEN"0110" =>dis_out<="0000010"; --6
20 | WHEN"0111" =>dis_out<="1111000"; --7
21 | WHEN"1000" =>dis_out<="0000000"; --8
22 | WHEN"1001" =>dis_out<="0010000"; --9
23 | WHEN"1010" =>dis_out<="0001000"; --A
24 | WHEN"1011" =>dis_out<="0000011"; --B
25 | WHEN"1100" =>dis_out<="1000110"; --C
26 | WHEN"1101" =>dis_out<="1000000"; --D
27 | WHEN"1110" =>dis_out<="0000110"; --E
28 | WHEN"1111" =>dis_out<="0001110"; --F
29 | WHEN OTHERS=>dis_out<="1111111";
30 | END CASE;
31 | END PROCESS;
32 | END fwm;

```

(6) 三个数码管 BCD8421 码输出

通过查询相关资料，资料来源见参考文献，这种问题的本质思想总结为八个字“移位乘二，大四加三”，移位乘二是高位数字从右往左，重新排列，大四加三是为了保证每位不超过九，因为十进制最大就是 9，经过我们逐步验算，我们还发现大四加三的循环是 BCD5421 码的排列。

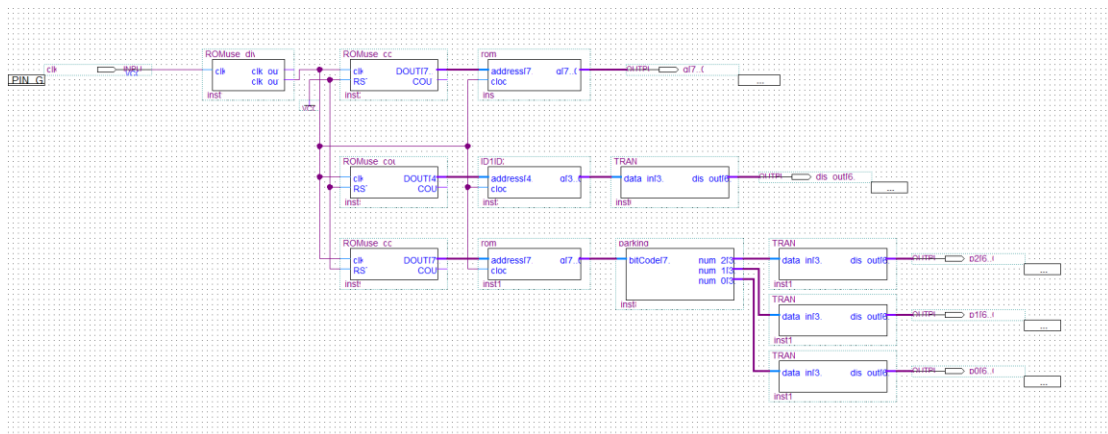
我们采用 Verilog 语言编写，相对简单。

```

1 module parkinglot(
2     input [7:0] bitCode,      // 输入信号, 8位二进制编码
3     output reg [3:0] num_2, num_1, num_0 // 输出信号, 3个4位二进制数
4 );
5
6 always @(bitCode)
7 begin
8
9     integer i;
10    integer num = 0; // 定义一个整数变量并初始化为0
11
12    num_2 = 4'b0; // 清零二进制数
13    num_1 = 4'b0; // 清零二进制数
14    num_0 = 4'b0; // 清零二进制数
15
16    for (i = 7; i >= 0; i = i - 1)
17    begin
18        //"大四加三"
19        if (num_2 >= 5) // 如果num_2大于等于5, 加上3, 限制范围在0-9
20            num_2 = num_2 + 3;
21        if (num_1 >= 5) // 如果num_1大于等于5, 加上3, 限制范围在0-9
22            num_1 = num_1 + 3;
23        if (num_0 >= 5) // 如果num_0大于等于5, 加上3, 限制范围在0-9
24            num_0 = num_0 + 3;
25        //移位乘2
26        num_2 = num_2 << 1; // 将num_2向左移动一位
27        num_2[0] = num_1[3]; // 将num_1的最高位赋值给num_2的最低位
28
29        num_1 = num_1 << 1; // 将num_1向左移动一位
30        num_1[0] = num_0[3]; // 将num_0的最高位赋值给num_1的最低位
31
32        num_0 = num_0 << 1; // 将num_0向左移动一位
33        num_0[0] = bitCode[i]; // 将输入信号的第i位赋值给num_0的最低位
34    end
35 end
36
37 endmodule

```

3.原理图

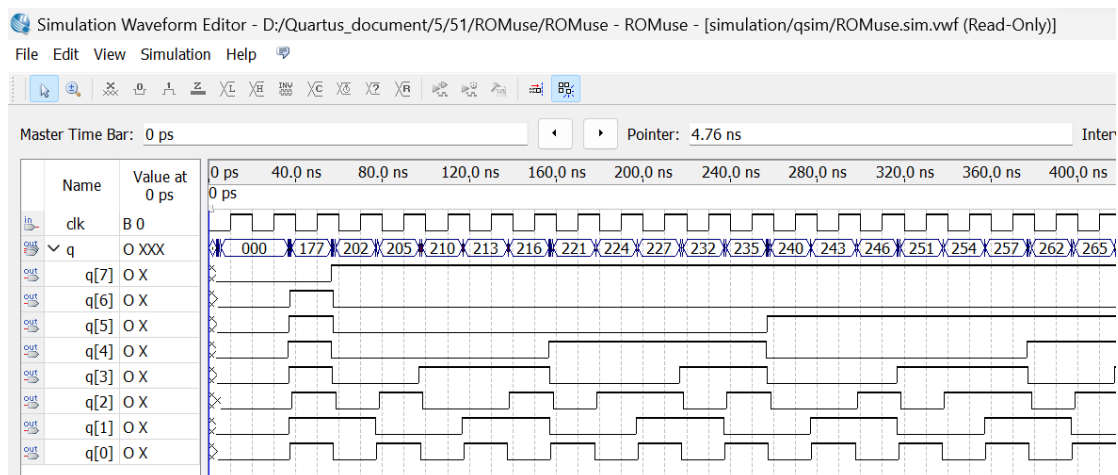


第一层实现正弦波数据 LED 灯显示;

第二层实现序列信号数码管显示;

第三层实现三个数码管分别显示正弦波数据的百位、十位、个位。

4. 波形仿真



5. 在实验开发板 DE0 上验证

以下只列出数码管显示，左边三个数码管显示正弦波，右边一个显示学号序列



经当堂验收，所有设计符合要求，上图仅呈现实验效果

六、实验过程中的问题

1. 对于三段数码管输出正弦波数据，通过提前预习，查询大量资料，总结得到。
2. 最开始 MIF 文件和 MATLAB 文件都尝试了，但只有 MIF 文件能 ROM 识别，走了些弯路。

七、心得体会

1. 从现在实验开始，内容基本上都是前面的综合，新的东西不多，需要你会灵活应用，具备快速学习的能力，比如移位加三法、Verilog 语言。ROM 在实现数据输出时和硬件编程语言一样比较简单。

2. 这次实验中我觉得最难得部分计算三个数码管 BCD8421 码输出。虽然难

懂，但通过查阅大量资料、和同学交流讨论，最终理解了为什么要移位和为什么加 3。

实验六 基于 FPGA 的信号发生器

一、实验目的

- 1、掌握锁相环（PLL）技术，学会用锁相环实现倍频功能；
- 2、熟悉 DAC 的功能特点，将 ROM 中的数字信号转成模拟信号；
- 3、复习示波器的使用方法。

二、实验要求

1.配置宽度为 8 位的 ROM，并在 ROM 中存储 256 个地址的正弦波数，用 PLL 生成 100M 时钟作为计数器计数脉冲，计数器输出作为地址读取 ROM 内容，将正弦波信号通过 DA 模块（AD9708）转换为模拟信号，通过示波器显示波形；

2.利用控制开关控制波形切换，比如当拨动开关置为 00 时，输出正弦波，当拨动开关置为 01 时，输入方波，当拨动开关置为 10 时，输出三角波，当拨动开关置为 11 时，输入锯齿波。

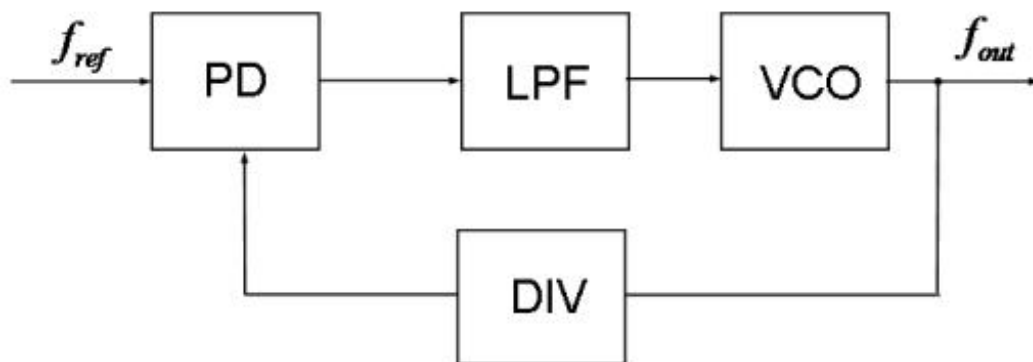
三、实验设备

- 1.Quartus II 13.0 软件；
- 2.FPGA 开发板；
- 3.AD-DA 模块一片；
- 4.函数发生器一台；
- 5.SMA-BNC 电缆一条；
- 6.杜邦线若干。

四、实验原理

1.PLL 锁相环

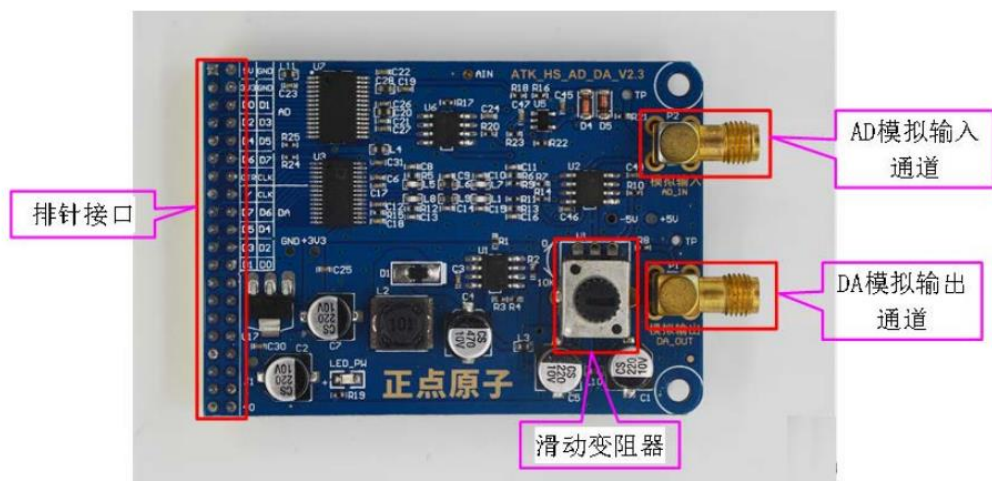
锁相环是一种利用反馈(Feedback)控制原理实现的频率及相位的同步技术,其作用是将电路输出的时钟与其外部的参考时钟保持同步。当参考时钟的频率或相位发生改变时,锁相环会检测到这种变化,并且通过其内部的反馈系统来调节输出频率,直到两者重新同步,这种同步又称为“锁相”。其原理图如下。



其中, PD 是鉴相器, LPF 是低通滤波器, VCO 是压控振荡器, DIV 是 DIV 分频器。

锁相环最重要的特征是,当环路锁定时,输入 PD 的两信号频率是严格相等的。这就派生出一种应用,就是在反馈回路上加上一个分频器。设其是 N 分频,则分频后信号频率等于输入信号频率;那么分频前,即环路输出信号的频率就一定是输入信号的 N 倍.这就是用锁相环实现倍频的原理。

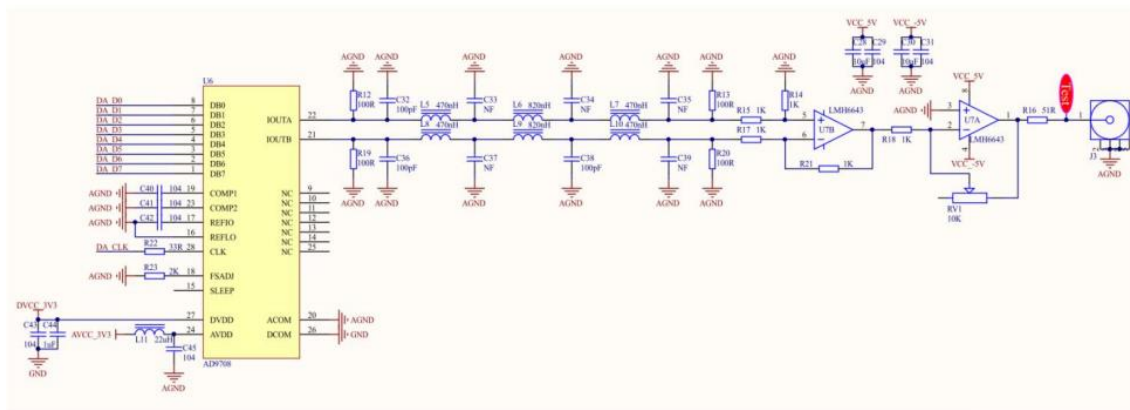
2.D/A 模块



3.AD9708 芯片

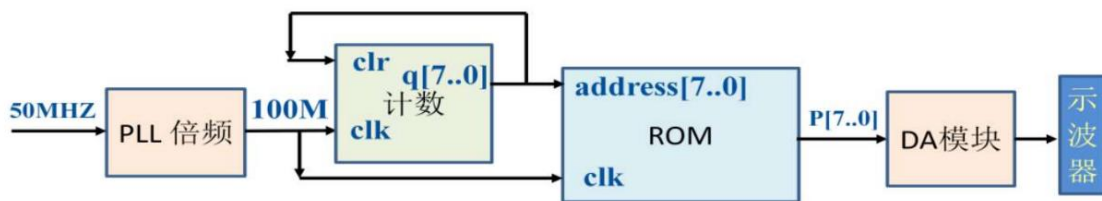
工作原理：AD9708 输出的一对差分电流信号先经过滤波器，再经过运放电路得到一个单端的模拟电压信号。图中 RV1 为滑动变阻器，可以调节输出的电压范围，使输出的电压范围在-5V 至+5V 之间。

原理图：



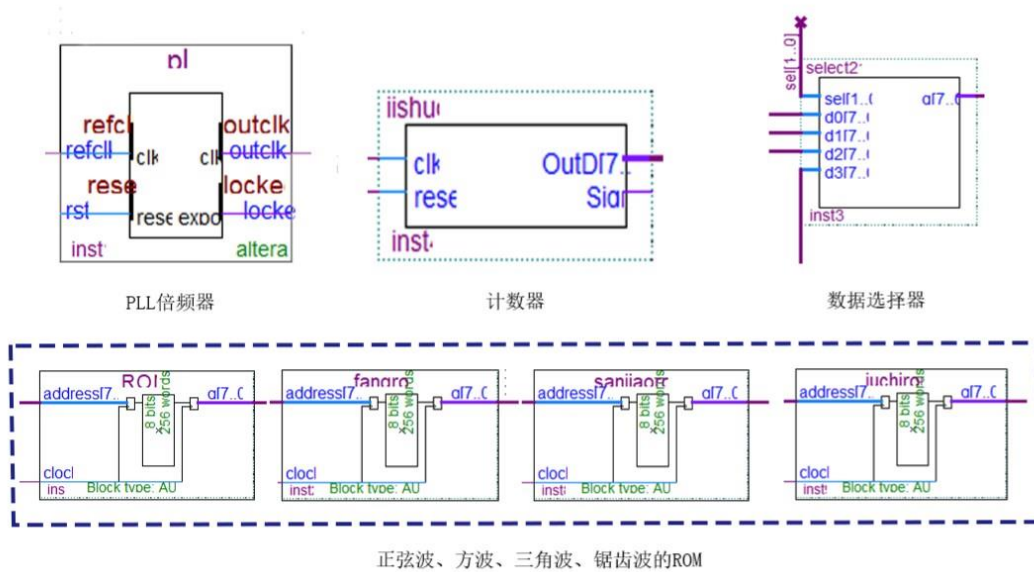
五、实验内容

1.实验原理图



本实验通过倍频器产生的时钟信号驱动计数器工作，计数模块扫描 ROM 中的数，通过 D/A 模块产生模拟信号，在示波器上显示。

2.模块配置



PLL 倍频器按照实验书制作完成。计数器和ROM均是前面的成果，代码略。

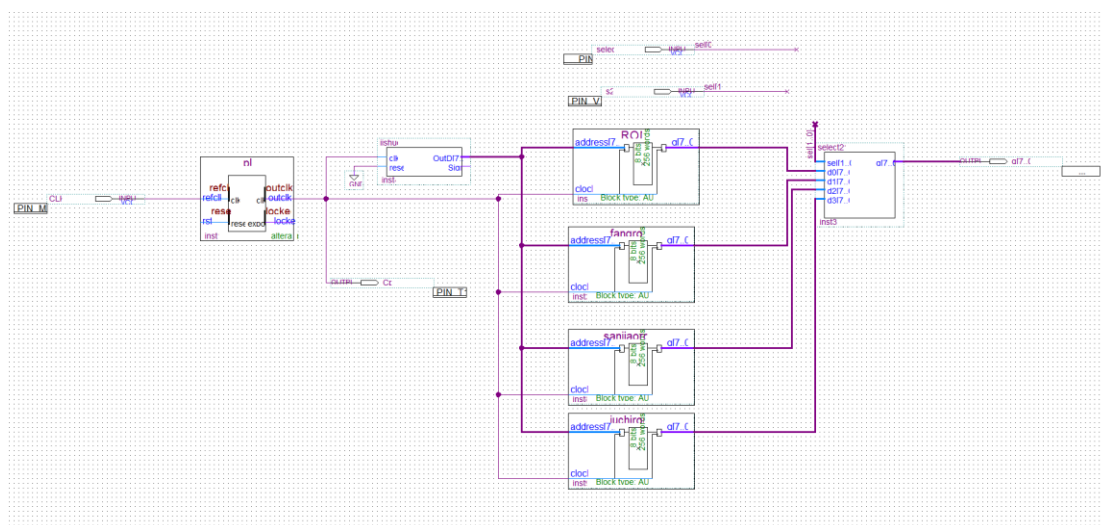
数据选择器的 VHDL 代码如下：

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  ENTITY select211 IS
6  PORT (
7      sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
8      d0, d1, d2, d3: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
9      q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
10 );
11 END select211;
12
13 ARCHITECTURE architecture OF select211 IS
14 BEGIN
15     PROCESS(sel)
16     BEGIN
17         CASE sel IS
18             WHEN "00" => q <= d0;
19             WHEN "01" => q <= d1;
20             WHEN "10" => q <= d2;
21             WHEN "11" => q <= d3;
22             --WHEN OTHERS => q <= d0; -- 默认选择d0
23         END CASE;
24     END PROCESS;
25 END architecture;

```

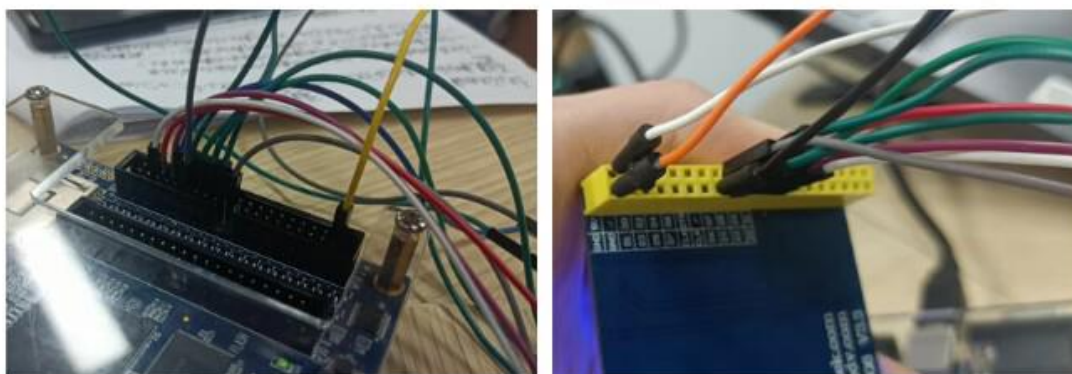
3.原理图



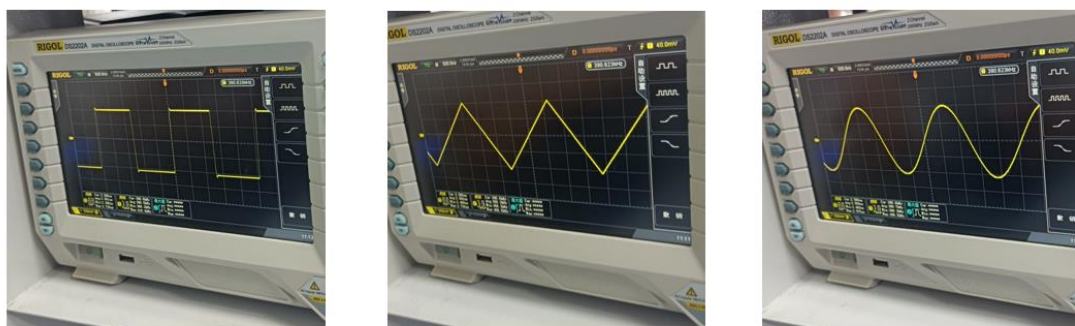
将以上模块组合，便得到如上的顶层文件原理图。

4.开发板验证

(1) D/A 模块连接



(2) 示波器显示

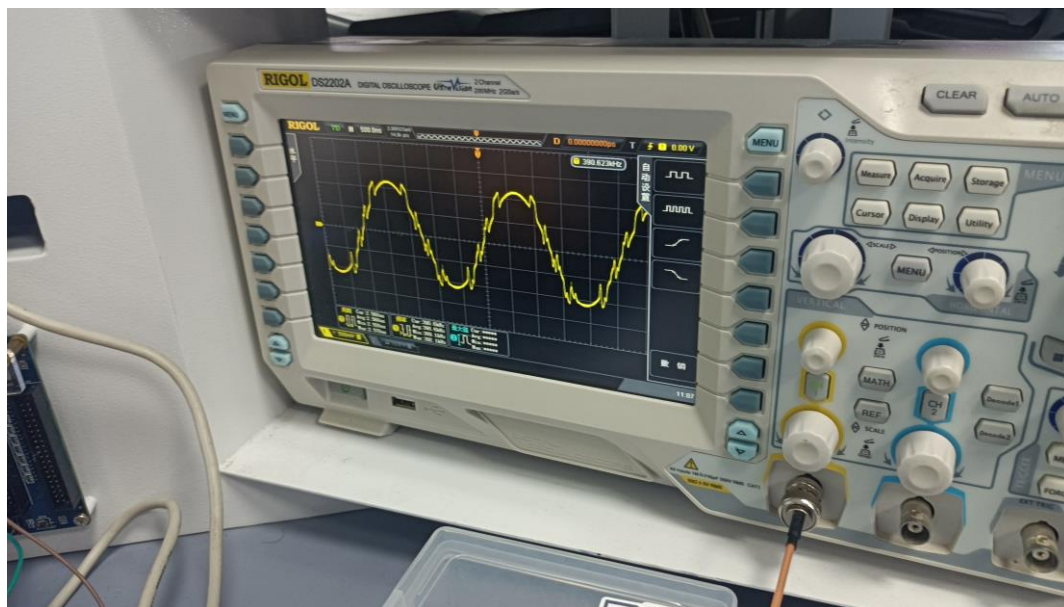


杜邦线连接处压紧之后，我们得到了清晰的波形，如上图所示。

六、实验过程中的问题

1.一开始显示的正弦波有毛刺锯齿，后来查明是杜邦线连接处接触不良所致。

实验波形如下图：



2.D/A 模块连接复杂，容易连错，耗时较长，其中有一次连接都对，却将 D/A 连到 A/D 处。

七、心得体会

1.通过这次实验了解了锁相环是一个方便获得所需频率的器件。我也对数模转换有了直观的硬件操作上的理解，加深了印象。

2.通过后面的几次试验，我逐渐探索系统工程文件的构建，目标分拆成几个模块，模块化设计是很重要的。

实验七 基于 FPGA 的模数转换电路

一、实验目的

- 1.熟悉模数转换电路的原理和功能特点，学会使用 A/D 转换模块；
- 2.用 FPGA 和 AD 转换模块实现检测方波高低电平的功能。

二、实验要求

1.使用 FPGA 开发板及高速 A/D 模块实现模数的转换。利用信号源输出频率为 1Hz 的方波信号，将方波信号输出连接至 A/D 模块的模拟电压输入端，A/D 模块将模拟信号转换成数字信号，并依次记录变化数据；

2.利用信号源产生频率 1Hz 方波，调节幅度旋钮，信号源输出信号幅度为+5V 至-5V 变化的信号时，当信号是+4V 输出时，2 位数码管输出显示组里同学学号后 2 位，当信号是-2.5V 输出时，2 位数码管输出显示组里另一位同学学号后 2 位；当信号源输出信号幅度为其他值时 2 数码管灭灯（不显示）。

三、实验设备

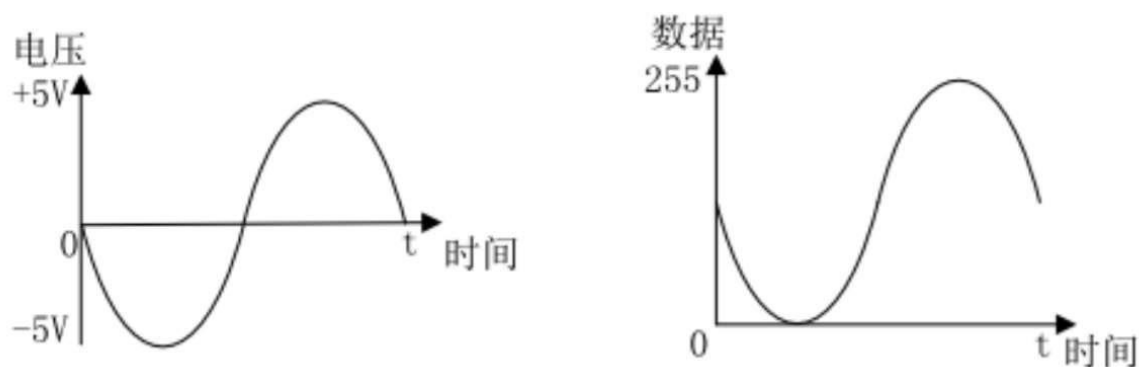
- 1.Quartus II 13.0 软件；
- 2.FPGA 开发板；
- 3.AD-DA 模块一片；
- 4.函数发生器一台；
- 5.SMA-BNC 电缆一条；
- 6.杜邦线若干。

四、实验原理

1.AD9280 芯片

AD9280 支持输入的模拟电压范围是 0V 至 2V，0V 对应输出的数字信号为 0，2V 对应输出的数字信号为 255。输出的电压范围是-5V~+5V，需要在 AD9280 的模拟输入端增加电压衰减电路，使-5V~+5V 之间的电压转换成 0V 至 2V 之间。那么实际上对我们用户使用来说，当 AD9280 的模拟输入接口连接-5V 电压时，AD 输出的数据为 0；当 AD9280 的模拟输入接口连接+5V 电压时，AD 输出的数据为 255。

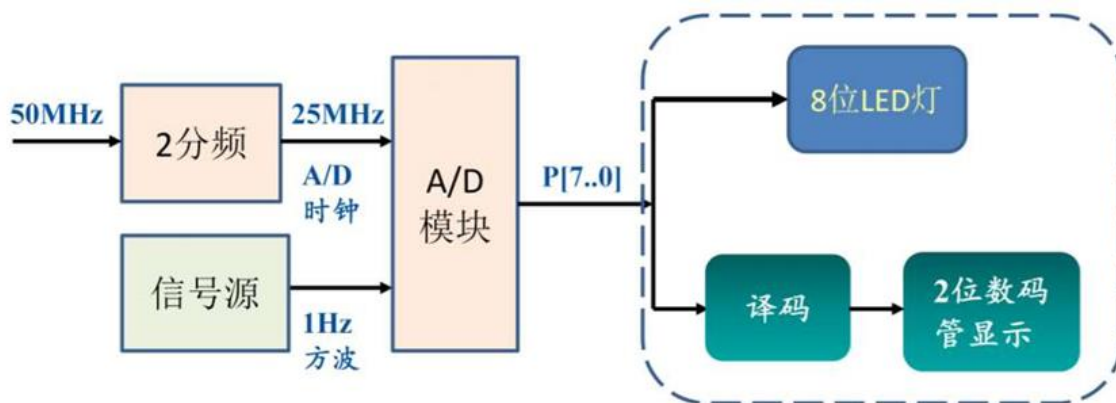
当 AD9280 模拟输入端接-5V 至+5V 之间变化的正弦波电压信号时，其转换后的数据也是成正弦波波形变化，转换波形如下图所示：



AD9280 正弦波模拟电压值（左）、数据（右）

五、实验内容

1.实验架构图

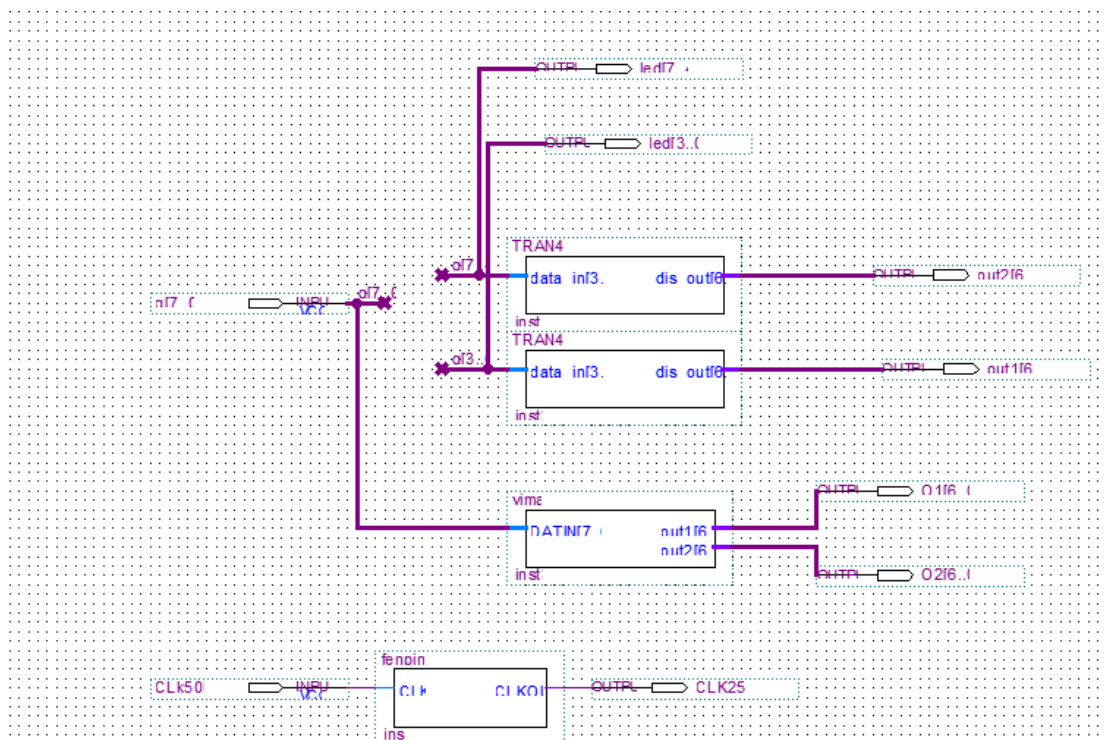


需要搭建的模块分为 2 分频模块、数码管译码器模块、幅值检测模块

2.模块构建



3.原理图



最上面为 LED 灯显示正弦波数据，最下面为二分频模块，中间为 16 进制的正弦波数据显示，这三个模块代码均是之前的代码，此处省略。

下面为幅值检测模块，VHDL 代码如下：

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity yima is
5  | port
6  | (
7  |   DATIN:in std_logic_vector(7 downto 0);
8  |   out1:out std_logic_vector(6 downto 0);
9  |   out2:out std_logic_vector(6 downto 0)
10 | );
11 | end yima;
12
13 | architecture xmk of yima is
14 | begin
15 |   process (DATIN)
16 |   | begin
17 |   | case DATIN is
18 |   |   when "11101010"=>out1<="1000000";out2<="0100100";
19 |   |   when "00111101"=>out1<="0100100";out2<="1111000";
20 |   |   when others=>out1<="1111111";out2<="1111111";
21 |   | end case;
22 |   | end process;
23 | end xmk;

```

4.开发板验证

(1) 利用信号源输出频率为 1Hz 的方波信号，将方波信号输出连接至 A/D 模块的模拟电压输入端，A/D 模块将模拟信号转换成数字信号，依次记录以下变化数据：

方波频率变化	正值	负值
+5V~5V变化	255	0
+4V~4V变化	232	22
+3V~3V变化	206	48
+2V~2V变化	183	76
+1V~1V变化	153	108
+2.5V~2.5V变化	193	61

下图为+5V~5V 变化的实验效果：



(2) 幅值检测

当信号是-2.5V (38) 输出时，2 位数码管输出显示 72；

当信号是+4V (E8) 输出时，2 位数码管输出显示 20；

当信号源输出信号幅度为其他值时 2 位数码管灭灯（不显示），如 00-FF 图所示。



经当堂验收，所有设计符合要求，上图只展示实验效果

六、实验过程中的问题

如上图所示，72 显示暗淡，是由于 38 左右波动太快，线又不紧，存在信号传输时差的问题，可能显示不明亮。

七、心得体会

这次实验结合了多次实验的学习内容，有一定难度。通过这次实验我对电路的模块化设计有更深入的了解。

参考文献

- [1]二进制转 BCD 码（8421）. <https://blog.csdn.net/XS20180801/article/details/84716098>
- [2]二进制转 BCD 码原理及 Verilog 实现. <https://blog.csdn.net/eroduandian123456/article/details/122551060>
- [3]多路彩灯控制器. <https://www.bilibili.com/video/BV1vo4y1X7Ya>