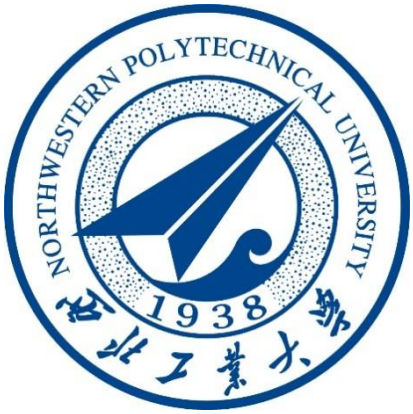


数字电子技术基础
实验报告

小组成员：2023302276 赵俊涛

小组成员：2023300573 孟文奇

组　　号：46

目录

实验一　TTL 集成门电路逻辑变换

一、实验目的

目的 1：熟悉 FPGA 的各接口与功能。 目的 2：熟悉 QuatusII13.0 的使用方法。 目的 3：回顾与非门的逻辑功能，实现一位全加器。

二、实验要求

要求 1.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.QuartusII13.0 软件； 2.FPGA 学习板。

四、实验原理

在将两个多位二进制数相加时，除了最低位以外，每一位都应该考虑来自低位的进位，即体两个对应位的加数和来自低位的进位 3 个数相加。这种运算称为全加，所用的电路称为全加器。

五、实验内容

1 .使用软件流程简介

创建工程项目

新建原理图文件

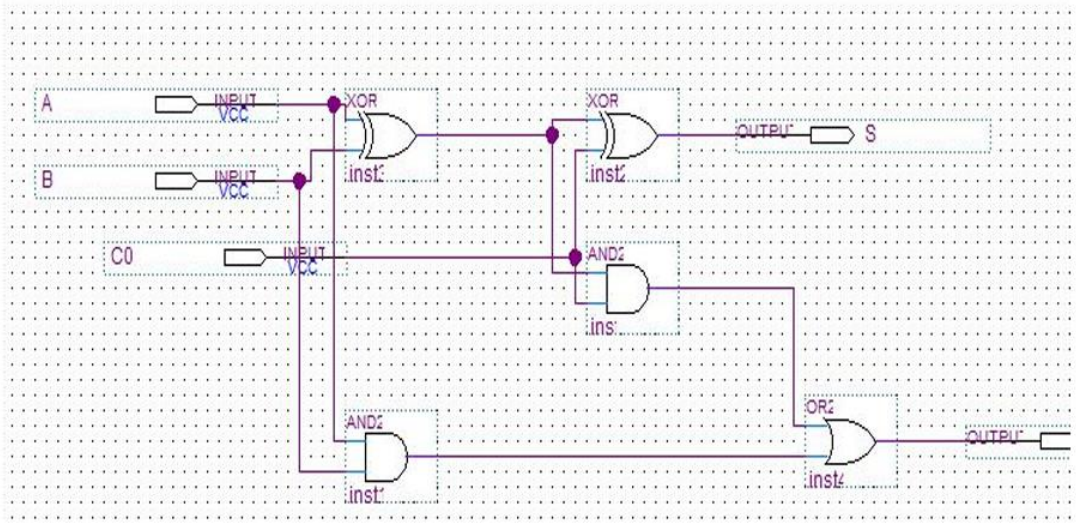
绘制原理图

编译程序

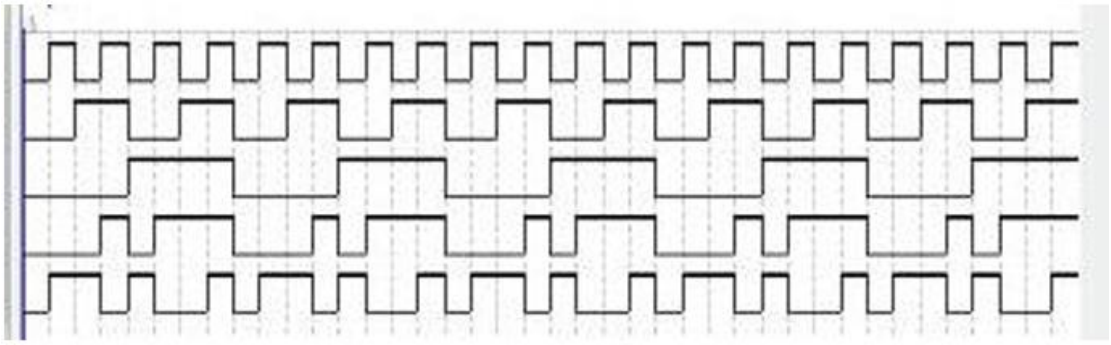
波形仿真

目标器件写入

2.全加器原理图



仿真结果：



六、实验过程中的问题

1.仿真软件使用不熟练，部分元器件的寻找与电路连接出现了一点麻烦。 **2.**工程文件创建时，文件的命名没有规律，妨碍了后续整理工作。

七、心得体会

1.相比直接搭建实物，使用仿真软件可以大大减少电路设计的工作量。

2.通过这次实验初步掌握了 quarts 软件的使用，将数电和硬件联系起来，很有意义。

实验二　组合电路设计

一、实验目的

目的 1：通过实验的方法学习数据选择器的电路结构和特点。

目的 2：掌握数据选择器的逻辑功能及其基本应用。

目的 3：通过实验的方法学习 74LS138 的电路结构和特点。

目的 4：掌握 74LS138 的逻辑功能及其基本应用。

二、实验要求

要求 1：参照参考内容，调用 MAXPLUSII 库中的组合逻辑器件 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。（QuartusII 实现波形仿真和下载开发板验证）

要求 2:参照参考内容，调用 MAXPLUSII 库中的组合逻辑器件 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。（QuartusII 实现波形仿真和下载开发板验证）

要求 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，其他情况，输出为 0）（QuartusII 实现波形仿真和下载开发板验证）

三、实验设备

1.QuartusII13.0 软件；

2.FPGA 学习板。

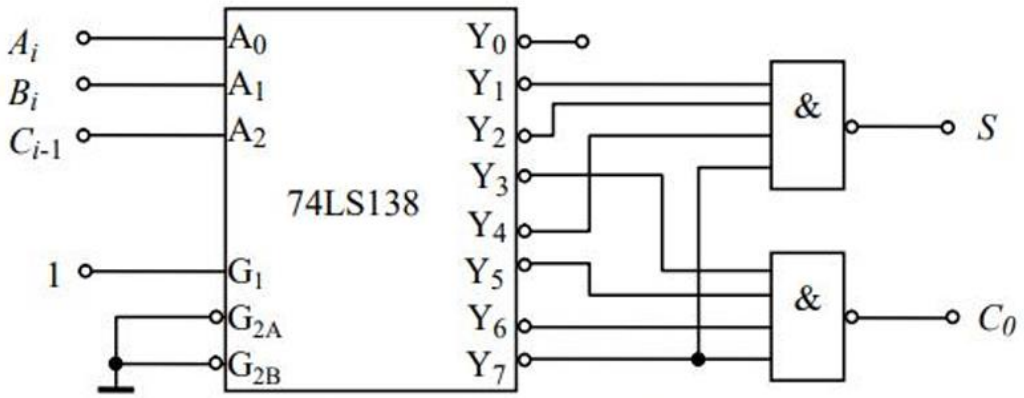
四、实验原理

1.用 74138 译码器实现一位全加器

真值表如下：

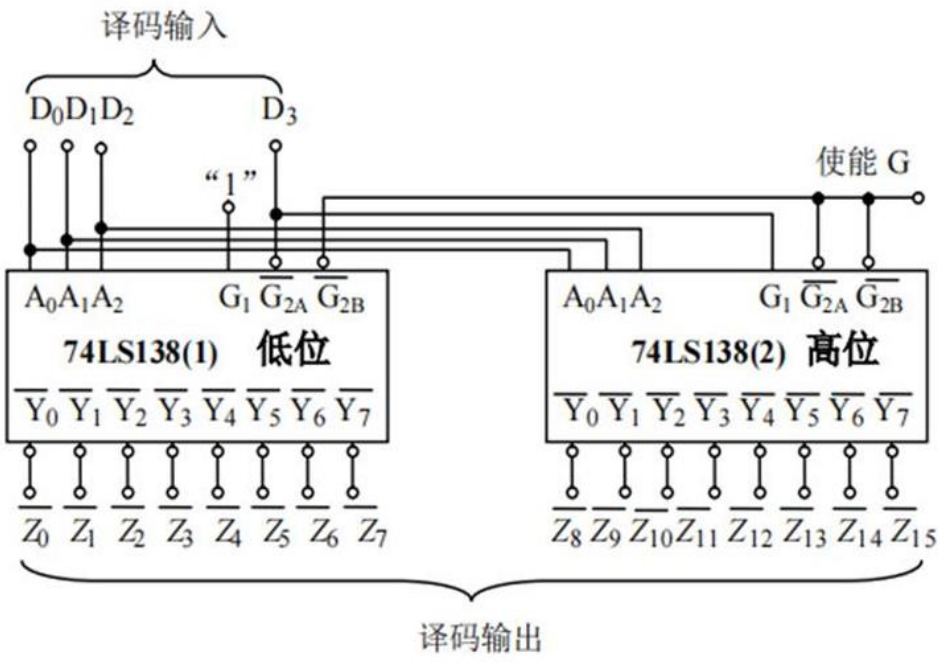
A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

而 74138 译码器每一个输出及对应一个最小项，故其原理图如下所示：



用 74LS138 实现全加器

2. 译码器、利用使能端将两片 74LS138（3/8 译码器）组合成一片四线十六线译码器



74LS138 扩展图

3.用 74153 实现一位全减器设计：

真值表为：

A1	B1	C1	D	CO1
0	0	0	0	0

0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

根据真值表画出卡诺图：

C1 \ A1B1	00	01	11	10
0	0	1	0	1
1	1	0	1	0

降维后：

B1 \ A1	0	1
0	C1	$\overline{C1}$
1	$\overline{C1}$	C1

C1	00	01	11	10
0	0	1	0	0

1	1	1	1	0

降维后：

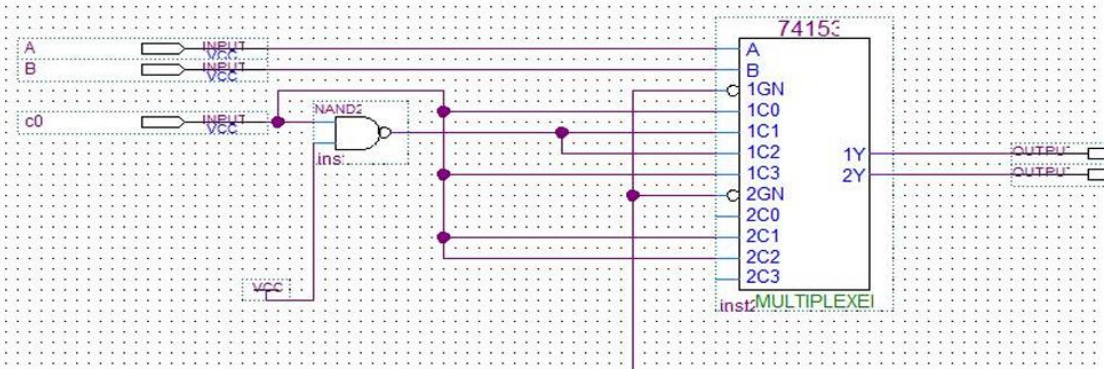
B1 \ A1	0	1
0	C1	0
1	1	C1

4、用 7447 数据选择器实现学号输入的检测

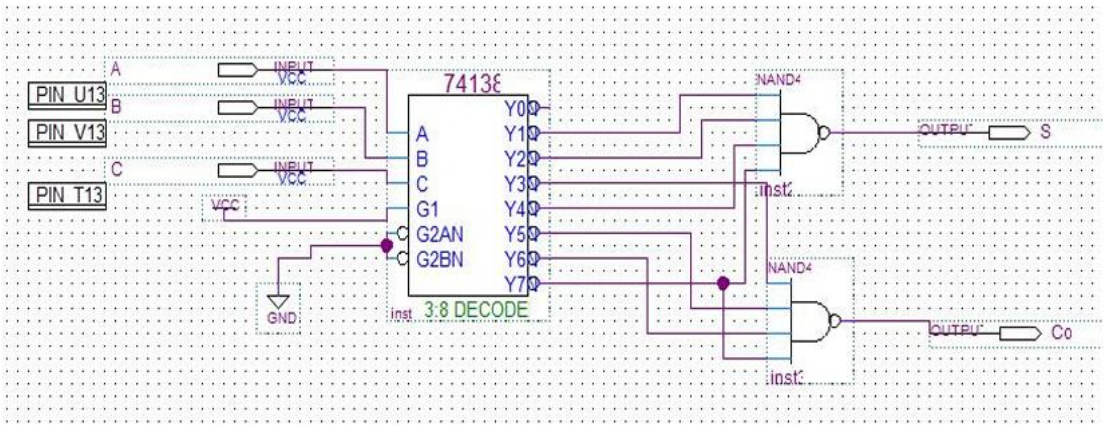
当开关控制端表示数字为“0（0000）”或“0（0000）”时，F 输出为 1， 表示进行后续全加器和全减器的使用；

五、实验内容

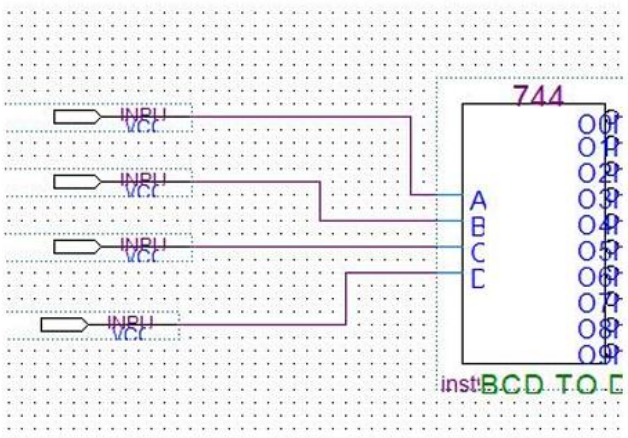
1. 一位全加器



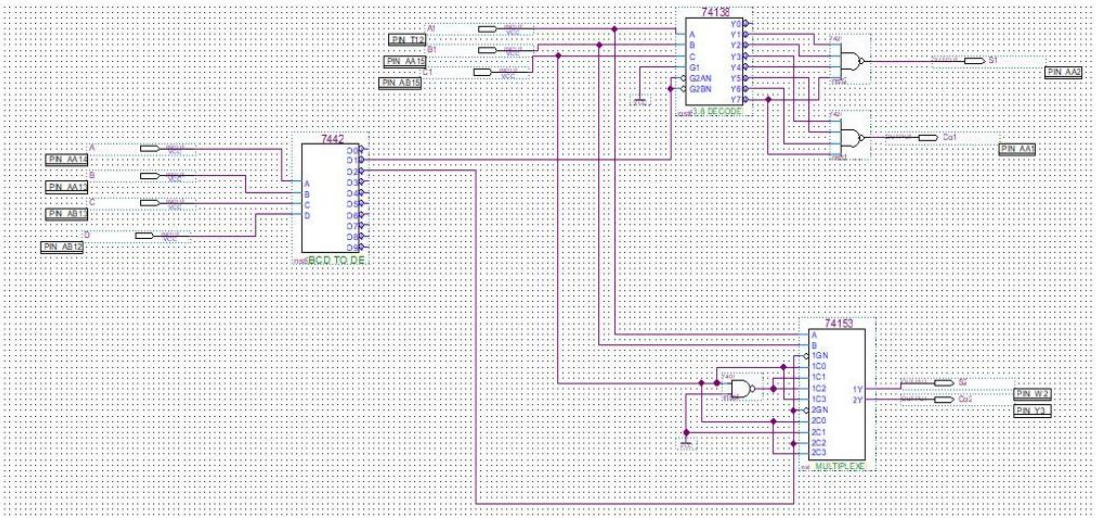
2. 一位全减器



3、用 7447 数据选择器实现学号输入的检测 当开关控制端表示数字为“0（0000）”或“0（0000）”时，F 输出为 1， 表示进行后续全加器和全减器的使用；



4.利用74LS138设计实现可选择的全加器和全减器 由于输出端：F(ABCD)= Σ(0,0)，因此当输入开关控制端 ABCD 表示数字为“0（0000）”或“0（0000）”时，F 输出为 1，表 示进行后续全加器和全减器的使用；最终实现，当输入学号为“0 “时，实现全加器，输入学号为”0“时，实现全减器；输入其 他数字，则不会实现任何功能。整体原理图如下:



六、实验过程中的问题

1.本次实验中需要根据真值表写出函数表达式，从而用译码器和双四选择器设计出电路，在设计过程中出现了电路连接错误的问题。

2.对组合电路的功能不熟悉，搭建电路时出现了一些问题。

七、心得体会

1.相比纯粹使用逻辑门，使用组合逻辑电路可以使电路设计模块化，增加通用性。

2.掌握了波形仿真的方法，有较大收获。

实验三 时序电路设计

一、实验目的

目的 1：通过实验的方法学习时序电路的基本设计方法。

目的 2：掌握时序电路的功能特点。

二、实验要求

要求 1:参照参考内容,用 QuartusII 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。

要求 2：参照参考内容中数码管显示控制电路设计方法，用 7490 二进制计数器、7447 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：学号后五位+ABCDE。

要求 3:参照参考内容,用 74161 二进制计数器、74194 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。

三、实验设备

1.QuartusII13.0 软件；

2.FPGA 学习板。

四、实验原理

1. DEO 开发板自带 50MHz 的时钟信号，但直接将其输出显示，频率太高肉眼无法分辨，因此需要将其分频为 1Hz 的信号。

2.7490 计数器是一种中规模二一五进制计数器。

3.7447 为译码器，使用共阳极接法，功能表如下。

Decimal or Function	Inputs						BI/RBO (Note 1)	Outputs							Note
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	(2)
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H	(3)
RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H	(4)

4. 74161 为异步清零式计数器，功能表如下。

5. 74194 为移位寄存器，功能表如下。

RD′	S1	S0	工作状态
0	*	*	置零
1	0	0	保持
1	0	1	右移
1	1	0	左移
1	1	1	并行输入

6. 学号的真值表为

Q_D	Q_C	Q_B	Q_A	D	C	B	A	结果
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0

0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	A
0	1	1	0	0	0	0	0	B
0	1	1	1	0	0	0	0	C
1	0	0	0	0	0	0	0	D
1	0	0	1	0	0	0	0	E

根据真值表画出卡诺图：

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

D 卡诺图

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X

10	0	0	X	X

C 卡诺图

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

B 卡诺图

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

A 卡诺图

7.彩灯设置 总体电路分为四大部分： 部分一由 Quartus 宏函数 lpm_counter 提供时钟脉冲信号； 部分二花型节拍控制电路由两片 74LS161 组成一个 32 进制计数器； 部分三花型演示电路由两片 74LS194 来控制花型； 部分四花型输出显示电路。

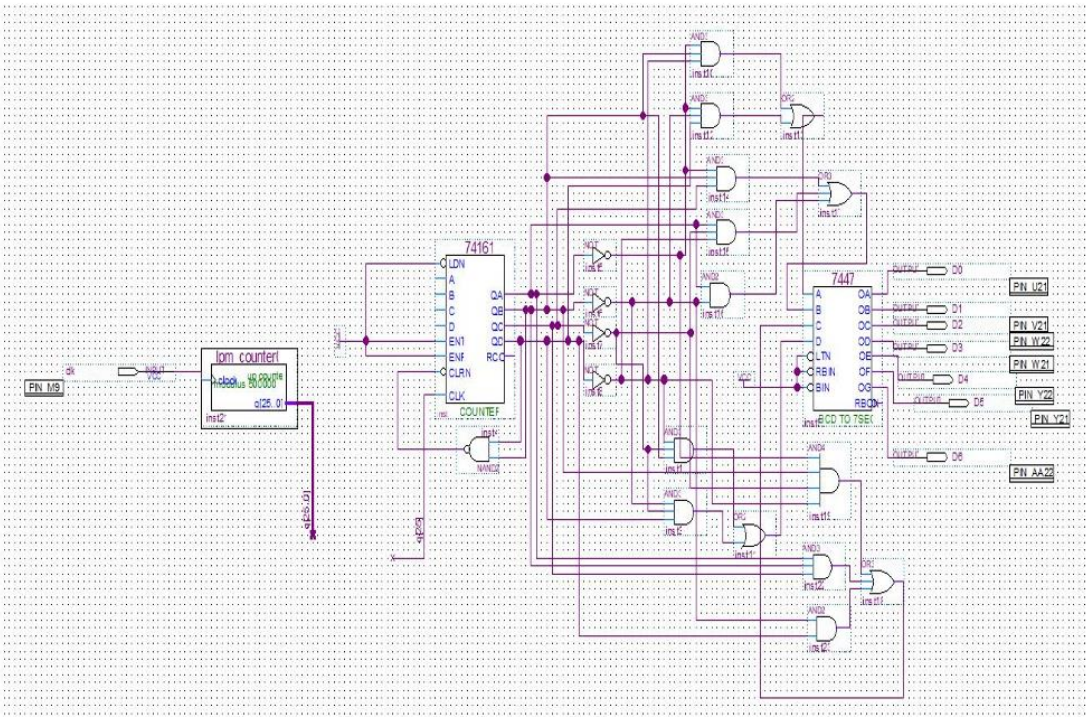
8 位彩灯分为 4 个一组，用两个 74LS194 来实现，花型Ⅰ——由两边向中间对称性依次亮，全亮后仍由两边向中间依次灭；花型Ⅱ——

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 两边对称性依次亮，全亮后仍由中间向两边依次灭，所以通过对花型 的分析可知，其中双向移位寄存器 74LS194(1)的功能是前 16 节拍 右移，后 16 节拍左移即先是 S1=0,S0=1,后变成 S1=1,S0=0，而 74LS194(2)则前 16 节拍为先左移后右移后 16 节拍也是先左移后右移。状态表部分如下所示，完整表格可参考文件：

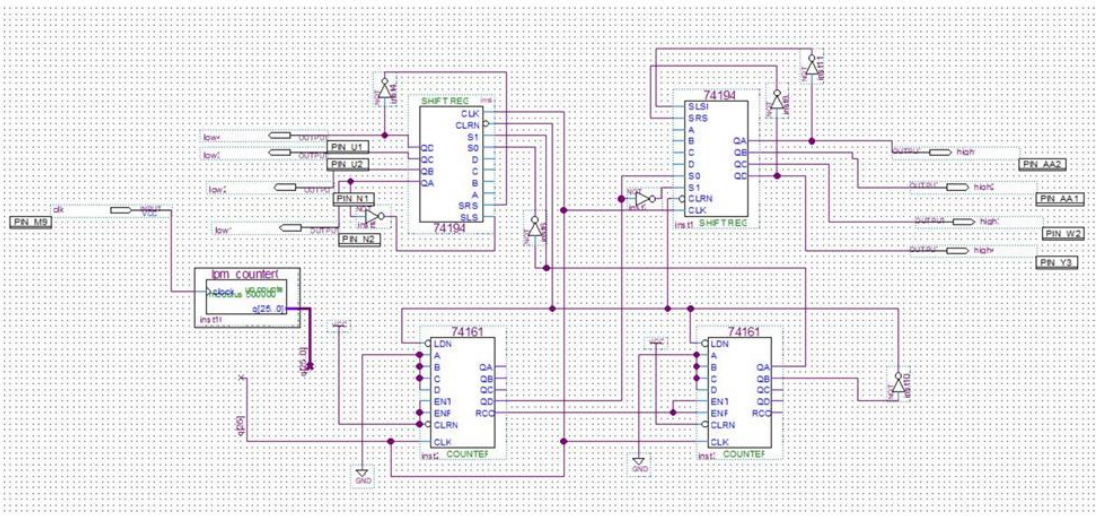
两片 74LS161		74LS194(1)			花型
74161（2）74161（1）		QA---QD	S1	S0	
QA	QDQCQBQA				
0	0000	1000	0	1	从左向右亮（花型Ⅰ）
0	0001	1100	0	1	从左向右亮（花型Ⅰ）
0	0010	1110	0	1	从左向右亮（花型Ⅰ）
0	0011	1111	0	1	从左向右亮（花型Ⅰ）

五、实验内容

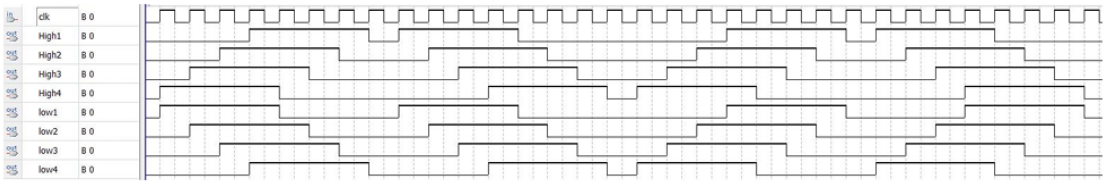
1.实现学号原理图如下。



2. 花灯实现原理图如下。



仿真结果如下：



六、实验过程中的问题

1.在实验中，我们出现了卡诺图绘制错误的问题(DCBA 对应 Q3 Q2 Q1 Q0 ， D 为高位 A 为低位)，结果没法输出正确的学号好在及时改正。

2.计算时容易将输入输出接口的顺序弄反，导致出错。

七、心得体会

1.在排线时一定要有所规划，这样不容易出错。

2.时序电路设计时，需要考虑电路的触发方式。

实验四 基于硬件描述语言电路设计

一、实验目的

目的 1：通过实验的方法学习硬件描述语言的基本使用方法。

目的 2：熟悉分频电路的功能特点。

二、实验要求

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。用 QuartusII 波形仿真验证；下载到 DE0 开发板验证。

要求 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。用 QuartusII 波形仿真验证；下载到 DE0 开发板，利用开发板

上的数码管验证。

要求 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 QuartusII 波形仿真验证；

要求 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 5M 和 50M 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 21 数字电子技术实验 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：

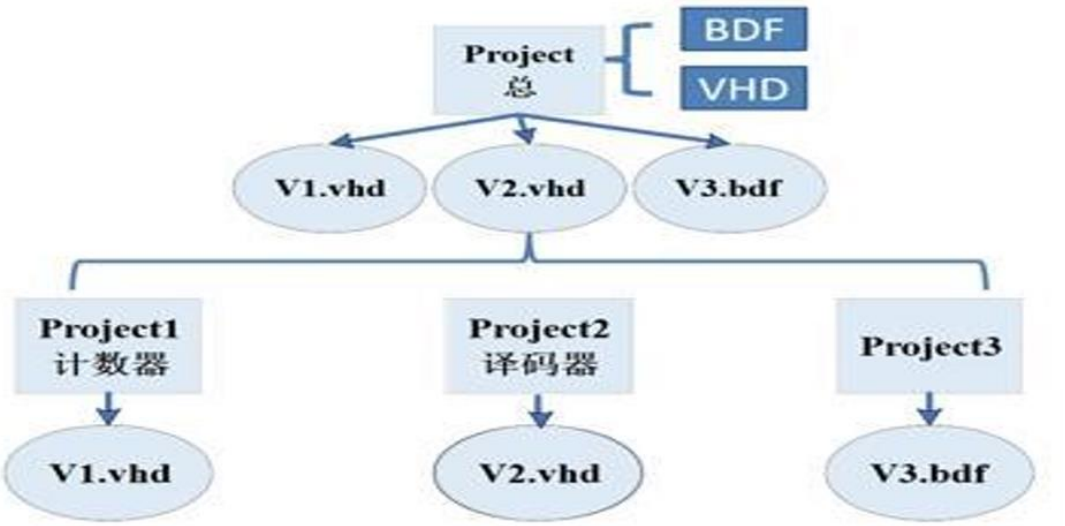
要求 5：利用已经实现的 VHDL 模块文件，顶层文件采用原理图设计方法，实现 A、b、C、d、E、学号 5 位、E、d、C、b、A 计数自动 循环显示，频率 1Hz 和 10Hz 可以切换。

二、实验设备

1.QuartusII13.0 软件；

2.FPGA 学习板。

四、实验原理



使用 VHDL 语言可生成用户所需特定功能的器件，相比于调用 quarts 软件原本的器件库,VHDL 语言模块性更强,有更好的灵活性。采用 VHDL 语言分别设计分频器、计数器、显示译码器,然后采用原 理图方式组合，形成最终电路。

五、实验内容

1.使用 VHDL 生成异或门。

代码：

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3  ⊟ENTITY sdsy_1 IS
4  ⊟PORT(A,B:IN STD_LOGIC;
5   ├C:OUT STD_LOGIC);
6   └END sdsy_1;
7  ⊟ARCHITECTURE fwm OF sdsy_1 IS
8  ⊟BEGIN
9   └C<=A xor B;
10  END;
```

2. 译码器

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3  ⊟Entity free_5 is
4  ⊟       PORT ( BCD_in  : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);    --输入四位BCD码
5  ├              SG_out  : OUT STD_LOGIC_VECTOR(6 DOWNTO 0));  --输出七位字形码
6  └ END;
7  ⊟ARCHITECTURE one OF free_5 IS
8  ⊟       BEGIN
9  ⊟        PROCESS(BCD_in)
10 │          BEGIN
11 ⊟           CASE  BCD_in  IS
12 │            WHEN "0000"  => SG_out <= "0001000";
13 │            WHEN "0001"  => SG_out <= "0000011";
14 │            WHEN "0010"  => SG_out <= "1000110";
15 │            WHEN "0011"  => SG_out <= "0100001";
16 │            WHEN "0100"  => SG_out <= "0000110";
17 │            WHEN "0101"  => SG_out <= "1000000";
18 │            WHEN "0110"  => SG_out <= "0110000";
19 │            WHEN "0111"  => SG_out <= "0100100";
20 │            WHEN "1000"  => SG_out <= "0000000";
21 │            WHEN "1001"  => SG_out <= "0010010";
22 │            WHEN "1010"  => SG_out <= "0000110";
23 │            WHEN "1011"  => SG_out <= "0100001";
24 │            WHEN "1100"  => SG_out <= "1000110";
25 │            WHEN "1101"  => SG_out <= "0000011";
26 │            WHEN "1110"  => SG_out <= "0001000";
27 │            WHEN OTHERS =>  NULL ;
28 ├           END CASE ;
29 └          END PROCESS;
30  END;
```

3. 分频器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY sdsy4_4 IS
PORT(clk:IN STD_LOGIC;
clk_out1:OUT STD_LOGIC;
clk_out2:OUT STD_LOGIC);
END sdsy4_4;
ARCHITECTURE fwm OF sdsy4_4 IS
CONSTANT m:INTEGER:= 2500000;
CONSTANT n:INTEGER:= 25000000;
SIGNAL tmp:STD_LOGIC;
SIGNAL tnp:STD_LOGIC;
BEGIN
PROCESS(clk,tmp)
VARIABLE cout : INTEGER:=0;
BEGIN
IF clk'EVENT AND clk='1' THEN
cout:=cout+1; --计数器+1
IF cout<=m THEN tmp<='0'; --计数小于等于 2500000，输出 0
ELSIF cout<m*2 THEN tmp<='1'; --计数小于 5000000，输出 1
ELSE cout:=0;
END IF;
```

```
IF clk'EVENT AND clk='1' THEN
cout:=cout+1; --计数器+1
IF cout<=m THEN tmp<='0'; --计数小于等于 2500000，输出 0
ELSIF cout<m*2 THEN tmp<='1'; --计数小于 5000000，输出 1
ELSE cout:=0;
END IF;
END IF;
END PROCESS;
clk_out1<=tmp;
PROCESS(clk,tnp)
VARIABLE cout : INTEGER:=0;
BEGIN
IF clk'EVENT AND clk='1' THEN
cout:=cout+1; --计数器+1
IF cout<=n THEN tnp<='0'; --计数小于等于 25000000，输出 0
ELSIF cout<n*2 THEN tnp<='1'; --计数小于 50000000，输出 1
ELSE cout:=0;
END IF;
```

```
IF cout<=n THEN tnp<='0'; --计数小于等于 25000000，输出 0
ELSIF cout<n*2 THEN tnp<='1'; --计数小于 50000000，输出 1
ELSE cout:=0;
END IF;
END IF;
END PROCESS;
clk_out2<=tnp;
END fwm;
```

4．计数器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY sdsy4_3 IS
    PORT ( clk,RST : IN STD_LOGIC;
           DOUT : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);   -- 四位计数
           COUT : OUT STD_LOGIC);                      -- 进位位
END sdsy4_3;
ARCHITECTURE fwm OF sdsy4_3 IS
SIGNAL Q1 : STD_LOGIC_VECTOR (3 DOWNTO 0);
BEGIN
PROCESS(clk,RST)
BEGIN
IF RST ='0' THEN Q1<=(OTHERS => '0'); COUT<='0';
ELSIF clk'EVENT AND clk='1' THEN Q1<=Q1+1; COUT<='0';
IF Q1 >= "1110" THEN Q1<=(OTHERS => '0'); COUT<='1';
END IF;
END IF;
END PROCESS;
DOUT<=Q1 ;
END fwm;
```

5.完整的电路图如下图所示。

六、实验过程中的问题

1.硬件描述语言 VHDL 比较难，需要较多的学习成本。

2.使用 VHDL 语言时，输入输出接口的设置需要特别注意。

1.
2.

七、心得体会

1．VHDL 语言相对于原理图输入法更加方便灵活。

2. 硬件描述语言运行时，部分语句并行执行，且不区分大小写与 C 语言有着很大区别。

实验五 FPGA 的 ROM（IP 核）使用

一、实验目的

目的 **1.** 熟悉 **MIF** 文件的生成方法。

目的 **2.** 熟悉 **ROM** 的功能特点。

二、实验要求

要求 1.调用 QuartusII 自带的 IP 核，生成 ROM，并设置 ROM 存储的数值，配置 数据位宽为 8 位的 ROM，并在 ROM 中存储 256 个地址的正弦波数据，利用计数器(如 74161 等)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。

要求 2.配置储存学号（同学 A+同学 B）的 ROM，利用计数器（如 74161 等)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 1 个七段数码管查看结果并验证。

三、实验设备

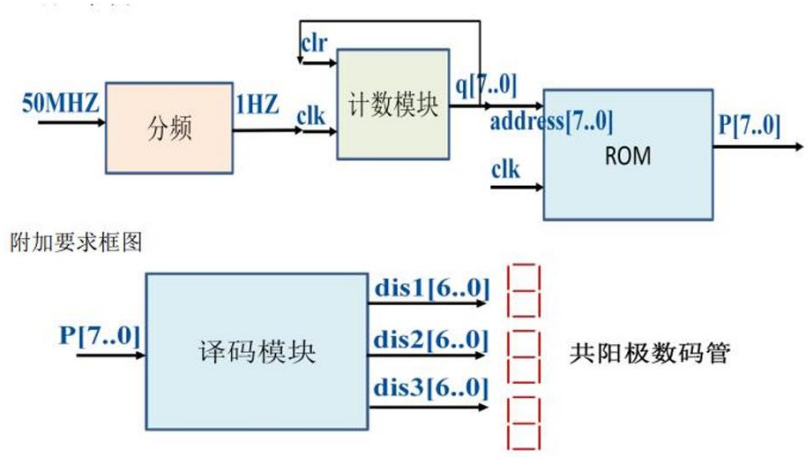
1.QuartusII13.0 软件；

2.FPGA 学习板。

四、实验原理

1. 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。

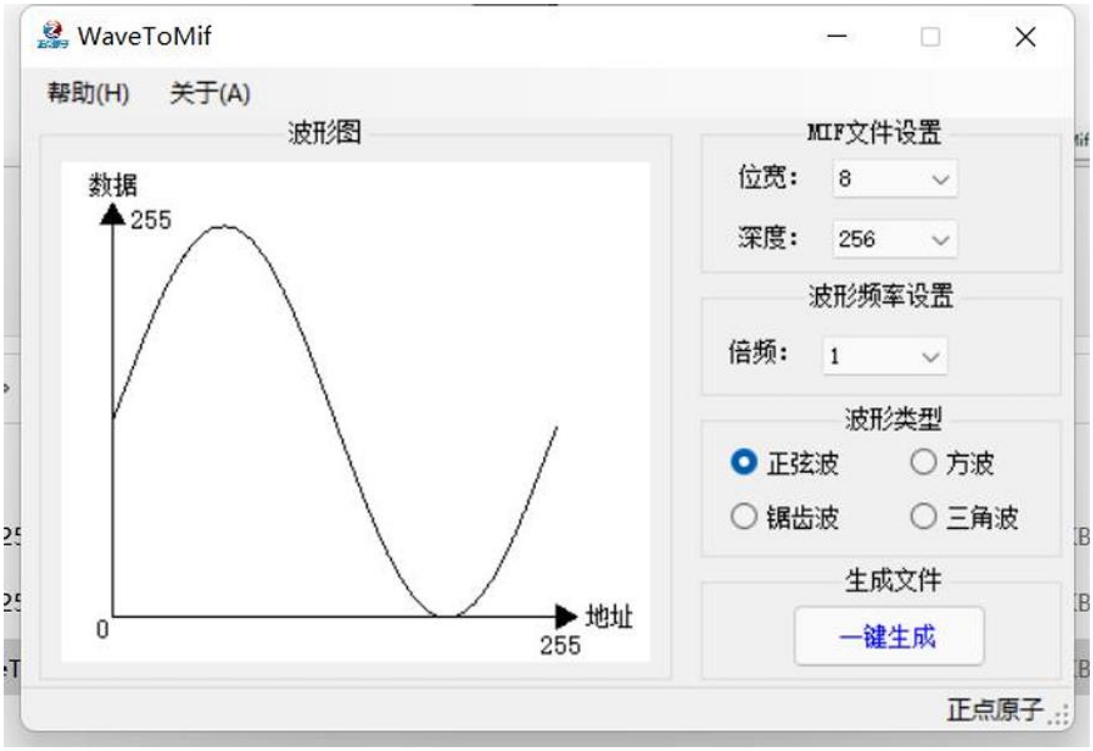
顶层框图：



五、实验内容

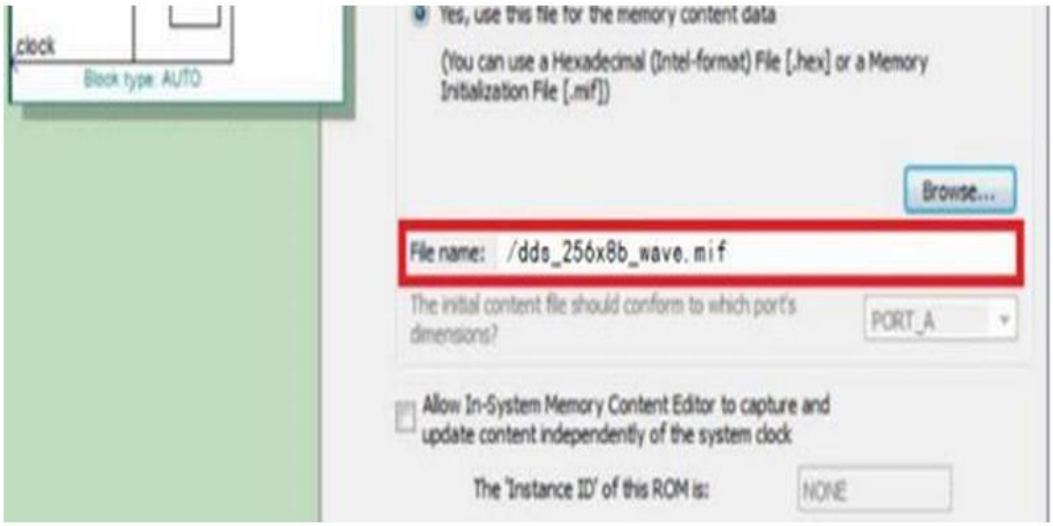
1. 波形数据生成

利用 MIF 软件（WaveToMif_V1.0.exe）生成，选择数据位宽、深度、倍频和波形类型（直接使用 MIF 软件生成是最直接的，还可使用 matlab 软件直接生成波形数据等）。

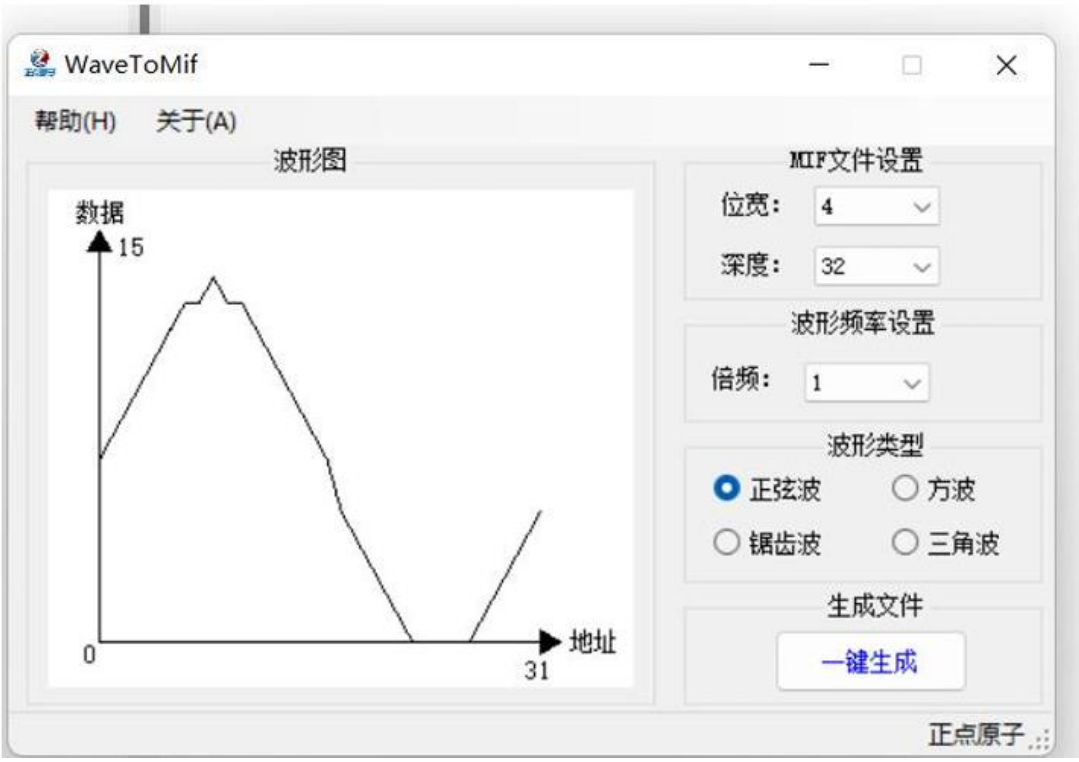



```
-- 波形转MIF上位机制作软件

WIDTH = 8;
DEPTH = 256;

ADDRESS_RADIX = UNS;
DATA_RADIX = UNS;

CONTENT BEGIN
    0   : 127;
    1   : 130;
    2   : 133;
    3   : 136;
    4   : 139;
    5   : 142;
    6   : 145;
    7   : 148;
    8   : 151;
```

4. 波形数据导入 ROM 模块。


clock
Block type: AUTO
Yes, use this file for the memory content data
(You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])
Browse...
File name: /dds_256x8b_wave.mif
The initial content file should conform to which port's dimensions?
PORT_A
Allow In-System Memory Content Editor to capture and update content independently of the system clock
The 'Instance ID' of this ROM is:
NONE

5. 学号ROM


WaveToMif
帮助(H)
关于(A)
波形图
数据
15
0
31
地址
MIF文件设置
位宽: 4
深度: 32
波形频率设置
倍频: 1
波形类型
正弦波
方波
锯齿波
三角波
生成文件
一键生成
正点原子

6. 分频器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY sdsy5_1hz IS
PORT(clk:IN STD_LOGIC;
clk_out1:OUT STD_LOGIC;
clk_out2:OUT STD_LOGIC);
END sdsy5_1hz;
ARCHITECTURE fwm OF sdsy5_1hz IS
CONSTANT m:INTEGER:= 2500000;
CONSTANT n:INTEGER:= 25000000;
SIGNAL tmp:STD_LOGIC;
SIGNAL tnp:STD_LOGIC;
BEGIN
PROCESS(clk,tmp)
VARIABLE cout : INTEGER:=0;
BEGIN
IF clk'EVENT AND clk='1' THEN
cout:=cout+1; --计数器+1
IF cout<=m THEN tmp<='0'; --计数小于等于 2500000，输出 0
ELSIF cout<m*2 THEN tmp<='1'; --计数小于 5000000，输出 1
ELSE cout:=0;
END IF;
END IF;
END PROCESS;
clk_out1<=tmp;
```

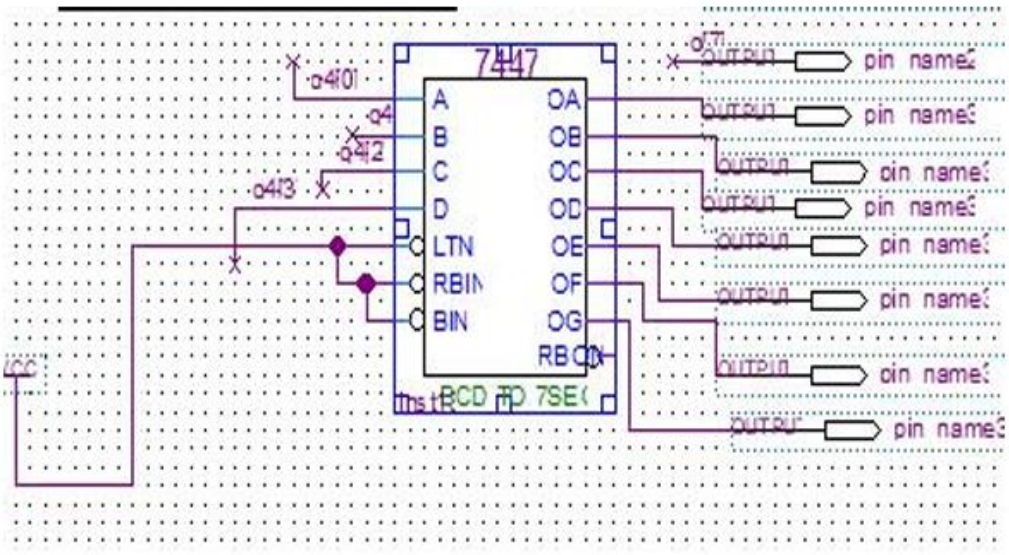
```
clk_out1<=tmp;
PROCESS(clk,tnp)
VARIABLE cout : INTEGER:=0;
BEGIN
IF clk'EVENT AND clk='1' THEN
cout:=cout+1; --计数器+1
IF cout<=n THEN tnp<='0'; --计数小于等于 25000000，输出 0
ELSIF cout<n*2 THEN tnp<='1'; --计数小于 50000000，输出 1
ELSE cout:=0;
END IF;
END IF;
END PROCESS;
clk_out2<=tnp;
END fwm;
```

7. 计数器

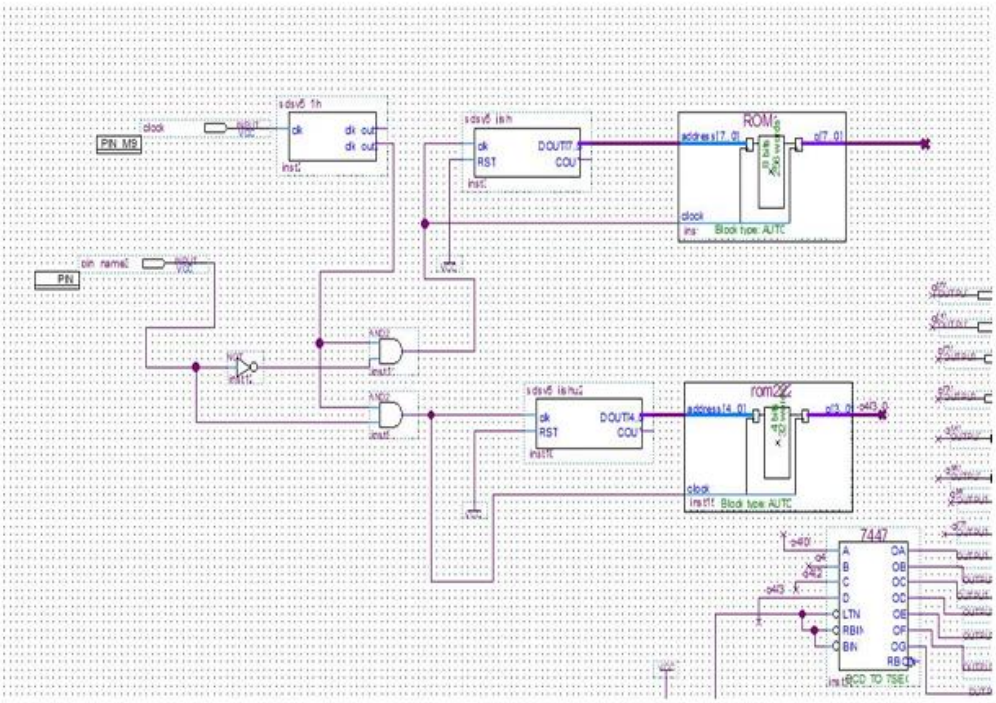
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY sdsy5_jishu IS
    PORT ( clk,RST : IN STD_LOGIC;
           DOUT : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);  -- 四位计数
           COUT : OUT STD_LOGIC);                     -- 进位位
END sdsy5_jishu;
ARCHITECTURE fwm OF sdsy5_jishu IS
SIGNAL Q1 : STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
PROCESS(clk,RST)
BEGIN
IF RST = '0' THEN Q1<=(OTHERS => '0'); COUT<='0';
ELSIF (clk'EVENT AND clk='1') THEN
Q1<=Q1+1;
COUT<='0';
IF Q1 >= "10000000" THEN Q1<=(OTHERS => '0'); COUT<='1';
END IF;
END IF;
END PROCESS;
DOUT<=Q1 ;
END fwm;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY sdsy5_jishu22 IS
    PORT ( clk,RST : IN STD_LOGIC;
           DOUT : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);  -- 四位计数
           COUT : OUT STD_LOGIC);                     -- 进位位
END sdsy5_jishu22;
ARCHITECTURE fwm OF sdsy5_jishu22 IS
SIGNAL Q1 : STD_LOGIC_VECTOR (4 DOWNTO 0);
BEGIN
PROCESS(clk,RST)
BEGIN
IF RST = '0' THEN Q1<=(OTHERS => '0'); COUT<='0';
ELSIF (clk'EVENT AND clk='1') THEN
Q1<=Q1+1;
COUT<='0';
IF Q1 >= "10100" THEN Q1<=(OTHERS => '0'); COUT<='1';
END IF;
END IF;
END PROCESS;
DOUT<=Q1 ;
END fwm;
```

8. 译码器



9. 整体原理图如下



六、实验过程中的问题

1.时钟信号会极大的影响电路的性能，要合理调节和选择。

2.ROM 设置的过程比较复杂，需要注意细节。

七、心得体会

1.通过这次实验对于存储的原理有了一定程度的认识。

2.使用软件更加熟练和灵活了。

实验六 基于 FPGA 的信号发生器

一、实验目的

目的 1.熟悉 MIF 文件生成方法，在实践中掌握锁相环技术。

目的 2.熟悉 DAC 的功能特点。

目的 3.熟悉示波器的使用方法。

二、实验要求

要求 1：用 PLL 生成 100M 时钟作为计数器计数脉冲，利用计数器（如 74161 等）或硬件描述语言生成的计数模块的输出依次扫描 ROM 的地址端读取 ROM 内容。

要求 2：配置宽度为 8 位的 ROM，并在 ROM 中存储 256 个地址的正弦波数据。

要求 3:将 ROM 输出的数字信号通过 DA 模块(AD9708)转换为模 拟信号，利用示波器显示波形。

三、实验设备

1.QuartusII13.0 软件；

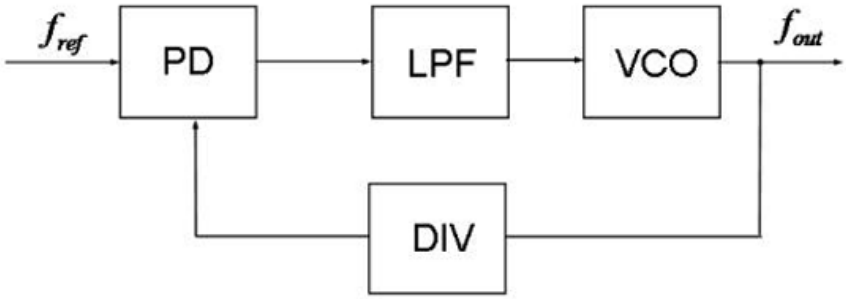
2.FPGA 学习板；

3.示波器；

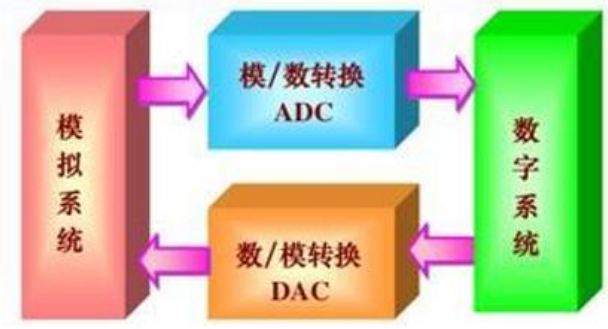
4. AD 模块。

四、实验原理

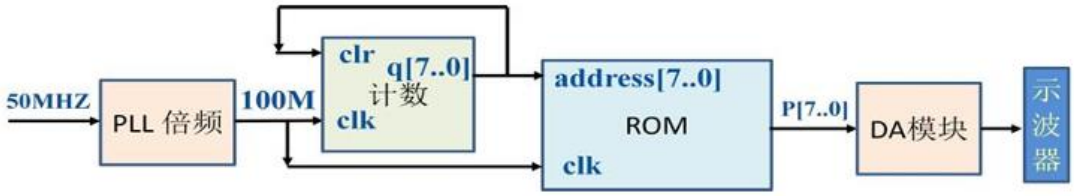
1. 锁相环（PLL:Phase-locked loops）是一种利用反馈（Feedback）控制原理实现的频率及相位的同步技术，其作用是将电路输出的时钟 与其外部的参考时钟保持同步。当参考时钟的频率或相位发生改变时， 锁相环会检测到这种变化，并且通过其内部的反馈系统来调节输出频 率，直到两者重新同步，这种同步又称为“锁相”。其原理图如下。



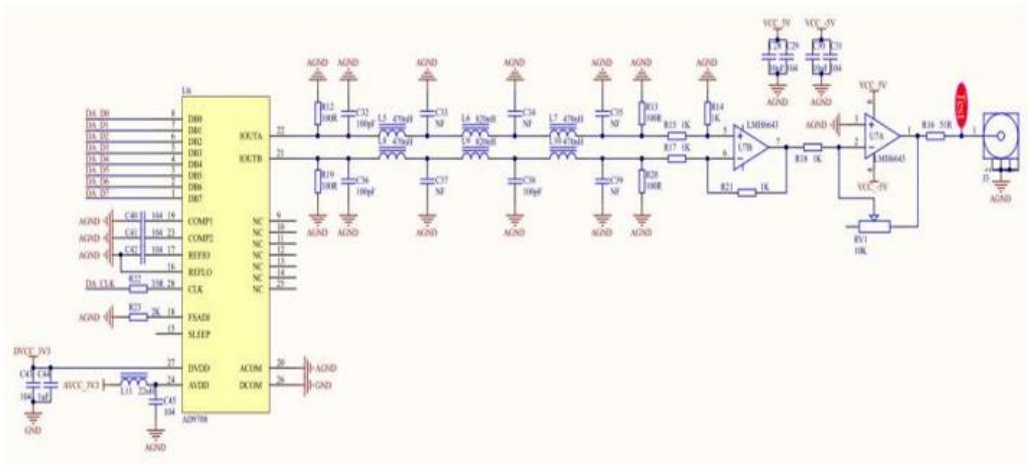
2.PLL 生成 100M 时钟作为计数脉冲，计数器输出作为地址读取 ROM 的内 容，经过 DA 模块转换为模拟信号，通过波形显示。



顶层框图如图所示：



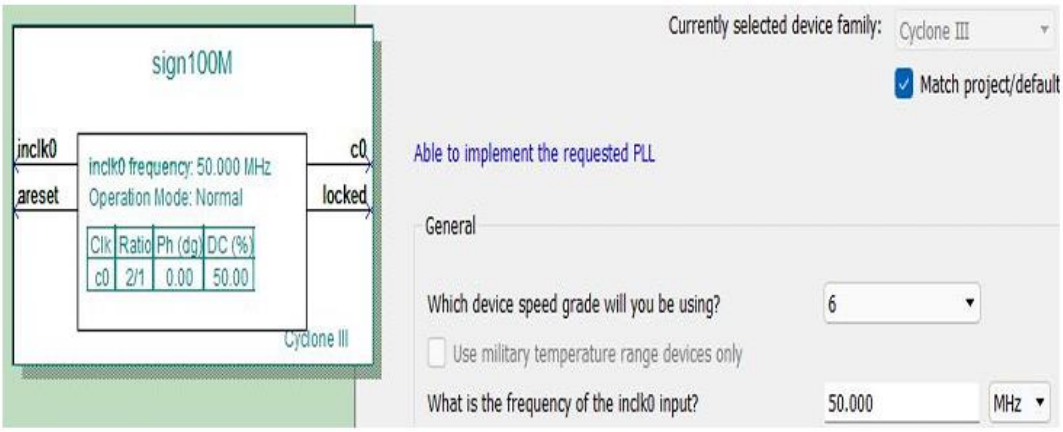
3.D/A 芯片 AD9708 原理图：



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。

五、实验内容

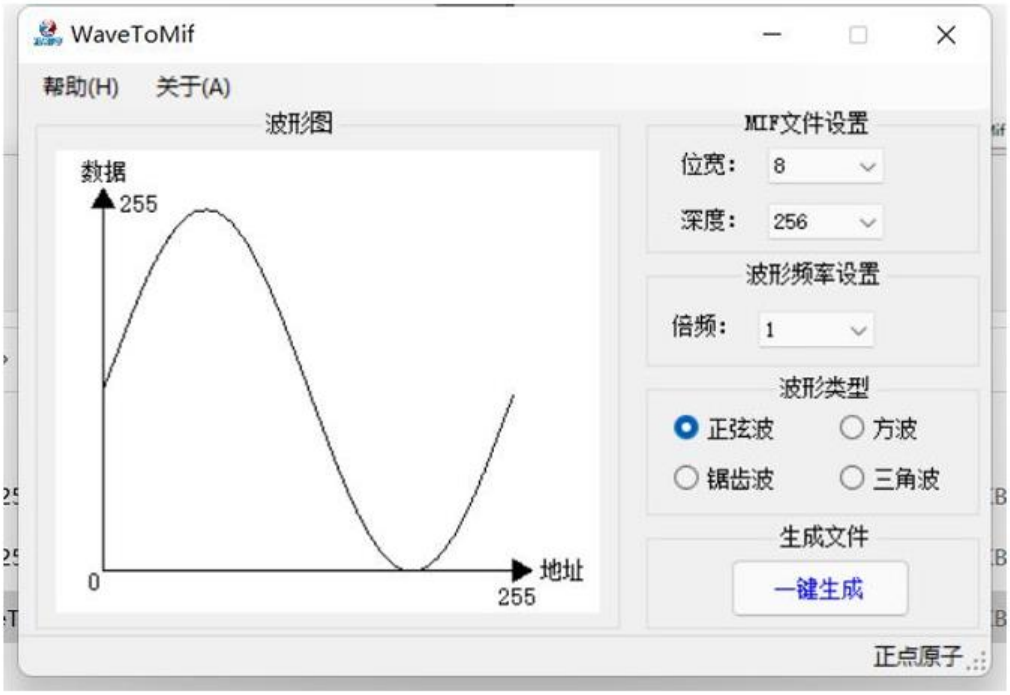
1.生成100M倍频器（具体过程可参考文件）

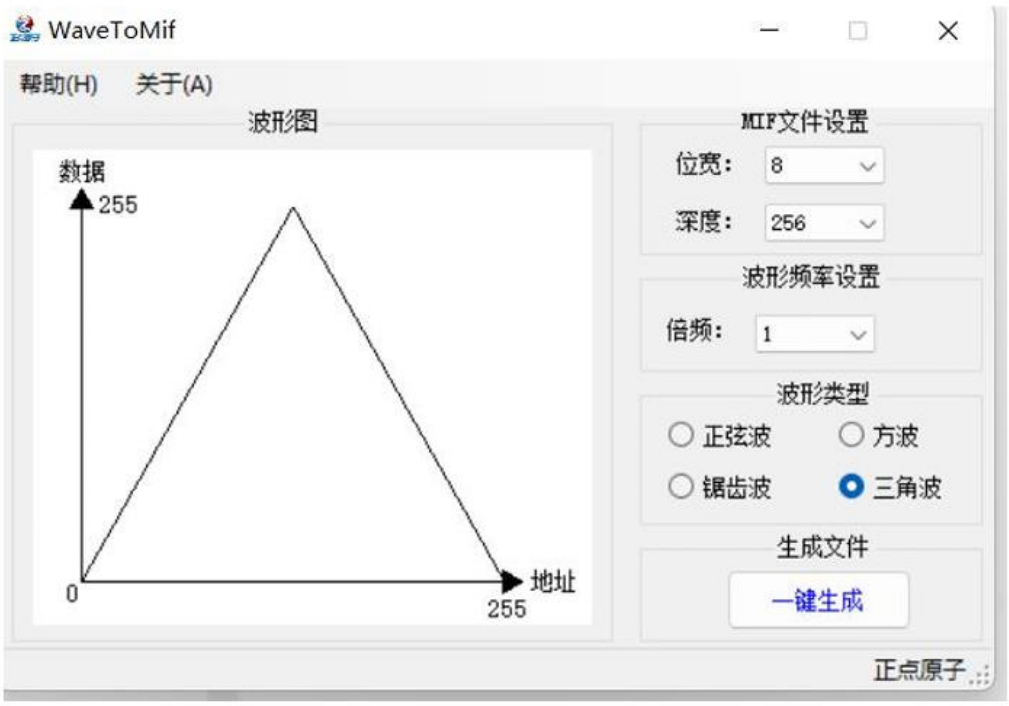


2. 计数器

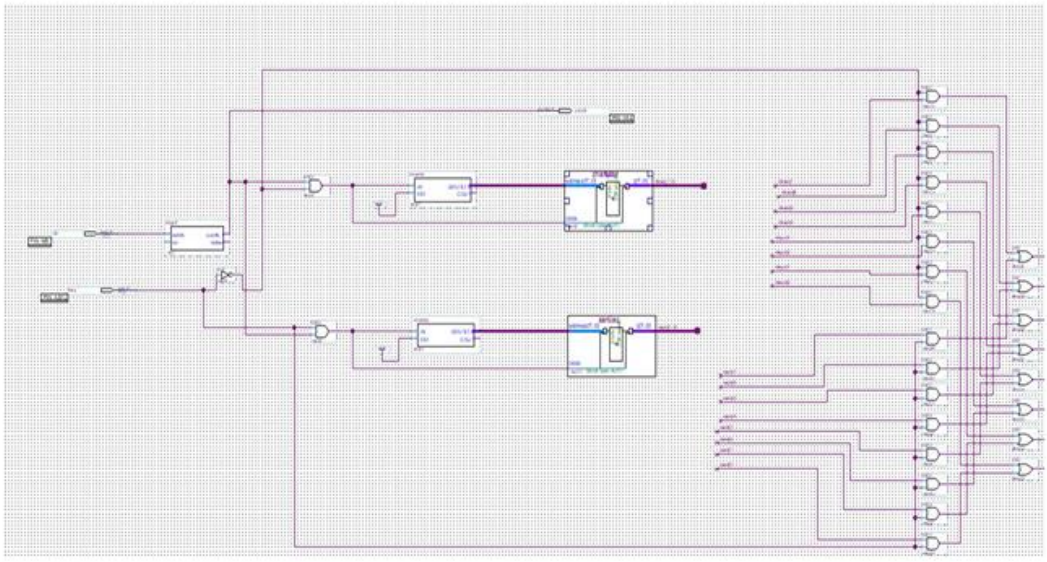
```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
4  ENTITY counter IS
5     PORT(clk,RST:IN STD_LOGIC;
6     DOUT:OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
7     COUT:OUT STD_LOGIC);
8  END counter;
9  ARCHITECTURE fwm OF counter IS
10    SIGNAL Q1:STD_LOGIC_VECTOR(7 DOWNTO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

3. ROM 的生成（正弦波和三角波）



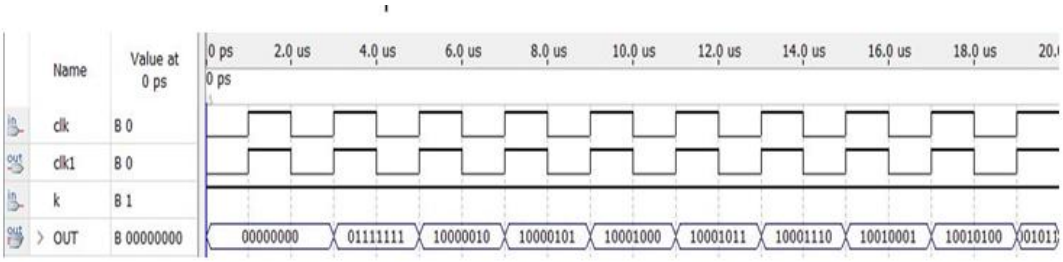


4. 得到原理图如下图所示

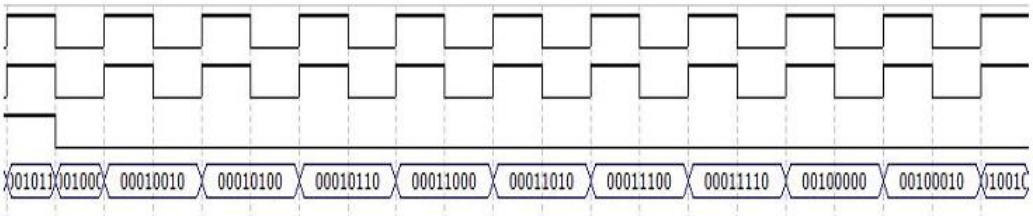


5.仿真结果如下图所示。

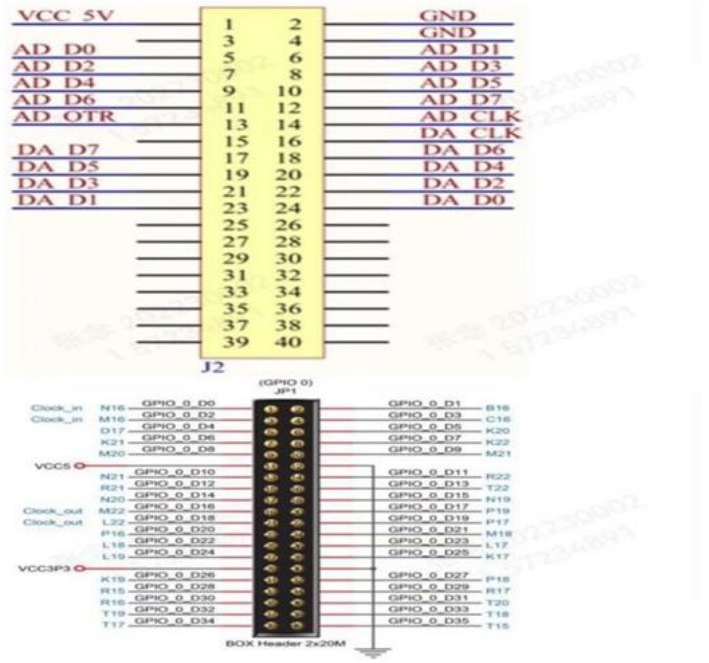
K 为高电平时实现正弦波:



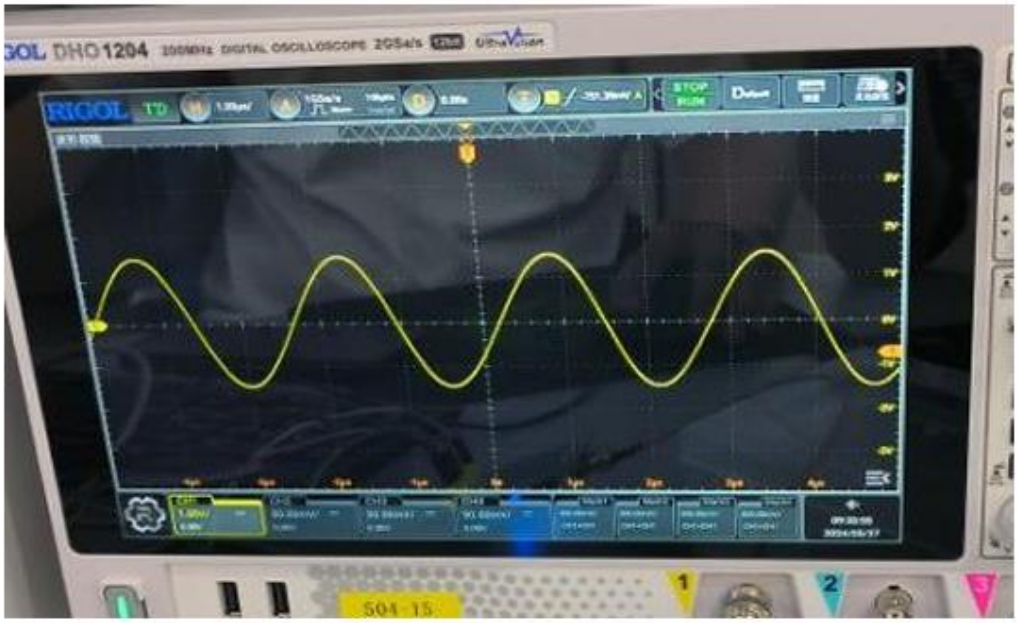
K 为低电平时实现三角波:

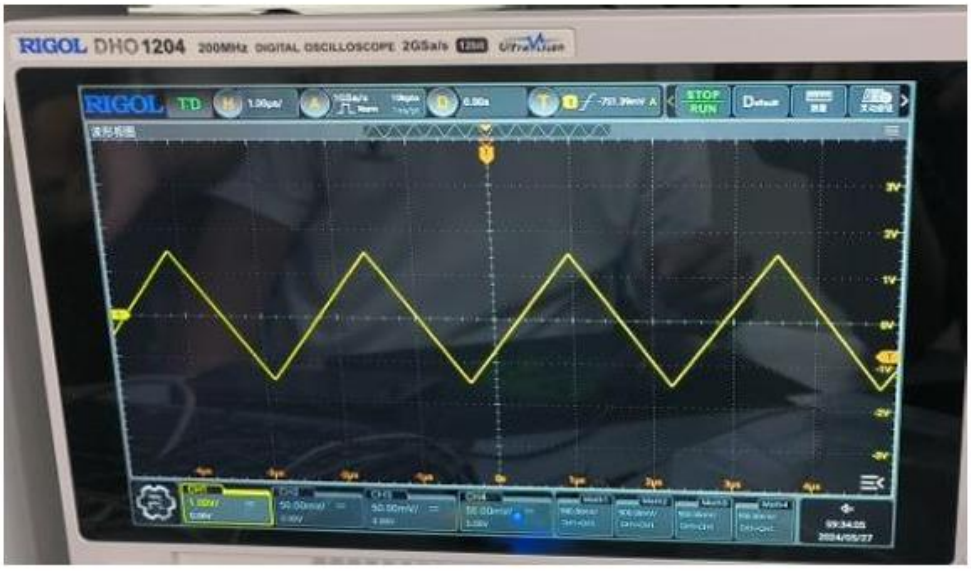


6.实物连接


VCC 5V
GND
AD D0
AD D1
AD D2
AD D3
AD D4
AD D5
AD D6
AD D7
AD OTR
AD CLK
DA CLK
DA D7
DA D6
DA D5
DA D4
DA D3
DA D2
DA D1
DA D0
J2
JP1
BOX Header 2x20M

7. 结果展示


DHO1204
RIGOL



六、实验过程中的问题

1.在实际的实验中，正弦波只显示下半个波形。

2.AD 模块和开发板之间连线时要特别注意，很容易连错，导致波形出错。

七、心得体会

1.通过这次的实验，让我对数字信号像模拟信号转换有了更加深入的了解。

2.锁相环技术可以很方便的获得需要的频率。

实验七　基于 FPGA 的模数转换电路

一、实验目的

目的 1. 熟悉 MIF 文件生成方法。

目的 2. 熟悉模数转换电路的功能特点。

二、实验要求

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。

要求 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+5 至-5V 变化的信号：信号是+5V 输出时，2 位数码 管输出显示组里同学学号后 2 位；信号是-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 位数码管灭灯（不显示）。

三、实验设备

1.QuartusII13.0 软件；

2.FPGA 学习板；

3.AD 模块；

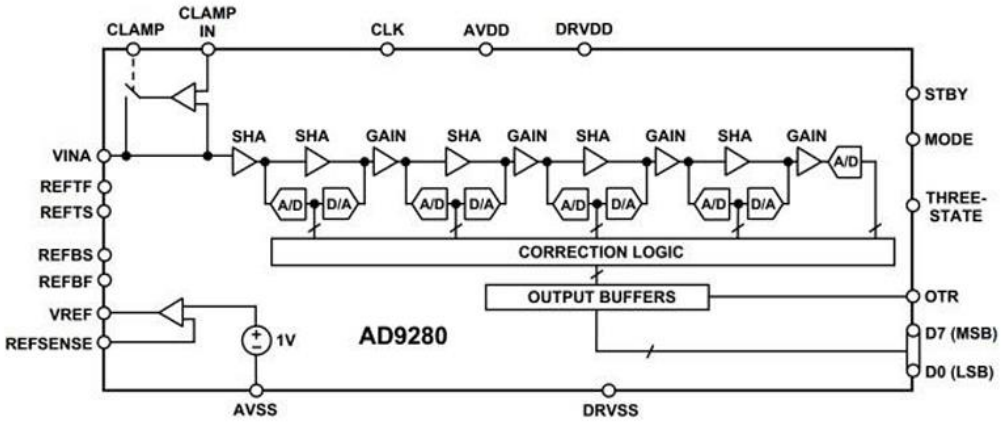
4.信号源。

四、实验原理

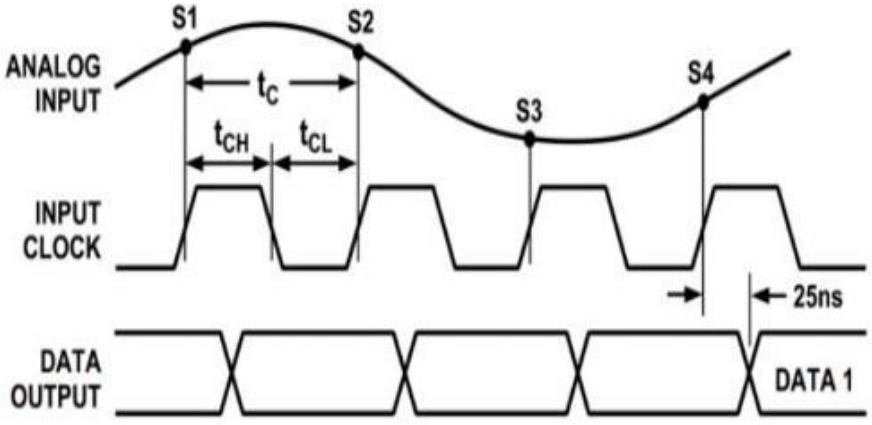
1.AD9280 模块

AD9280 是 ADI 公司生产的一款单芯片、8 位、32MSPS（Million Samples Per Second，每秒采样百万次）模数转换器，具有高性能、低功耗的特点。

AD9280 的内部功能框图如下图：



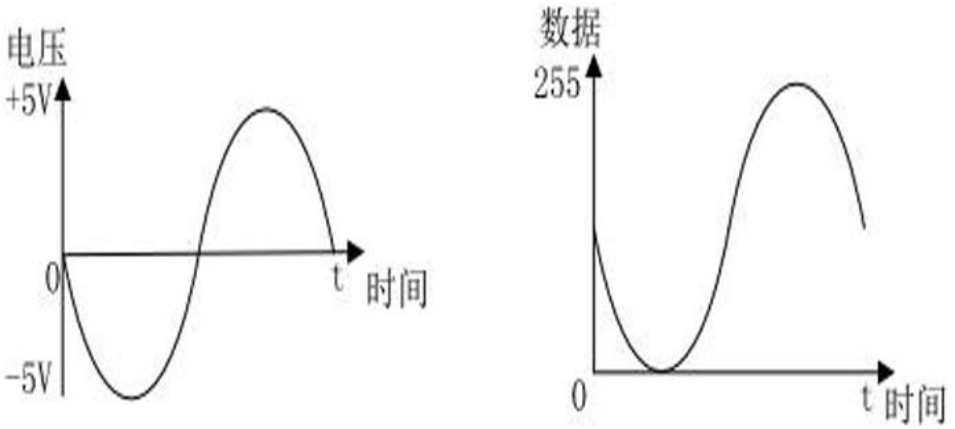
AD9280 在时钟(CLK)的驱动下工作,用于控制所有内部转换的周期。AD9280 输出的数据以二进制格式表示，当输入的模拟电压超出量程时，会拉高 OTR（out-of-range）信号；当输入的模拟电压在量程范围内时，OTR 信号为低 电平，因此可以通过 OTR 信号来判断输入的模拟电压是否在测量范围内。



AD9280 时序图

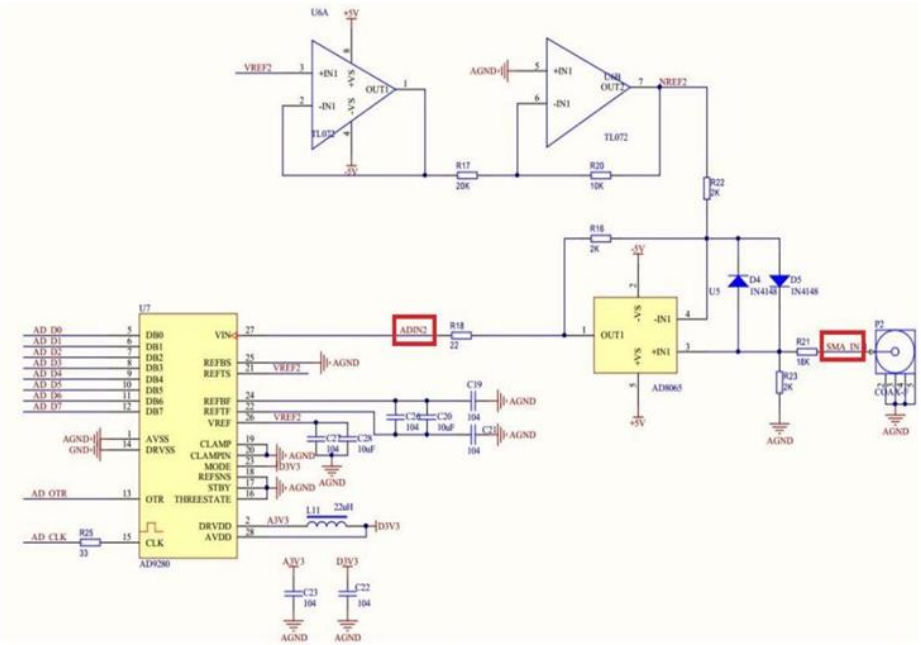
需要注意的是，AD9280 芯片的最大转换速度是 32MSPS，即输入的时钟最 大频率为 32MHz。AD9280 支持输入的模拟电压范围是 0V 至 2V，0V 对应输出的数字信号为 0，2V 对应输出的数字信号为 255。输出的电压范围是-5V~+5V，需要在 AD9280 的模拟输入端增加电压衰减电路，使-5V~+5V 之间 的电压转换成 0V 至 2V 之间。那么实际上对我们用户使用来说，当 AD9280 的 模拟输入接口连接-5V 电压时，AD 输出的数据为 0；当 AD9280 的模拟输 入接口连接+5V 电压时，AD 输出的数据为 255。

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：

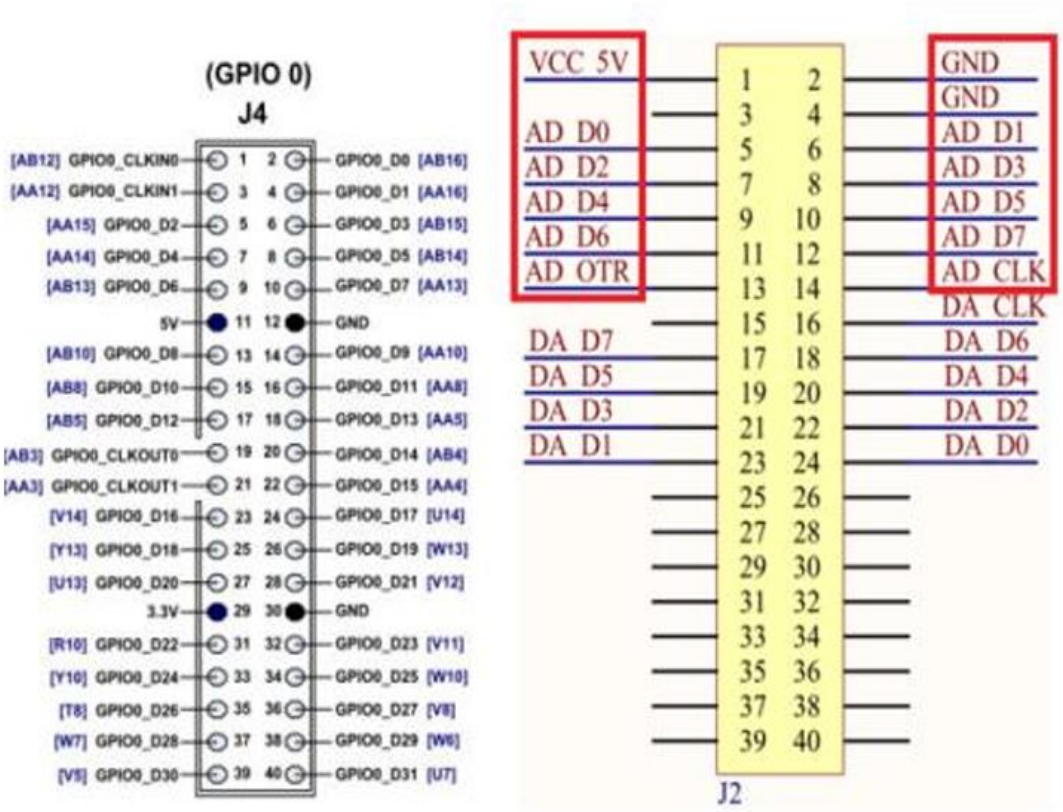


AD9280 正弦波模拟电压值（左）、数据（右）

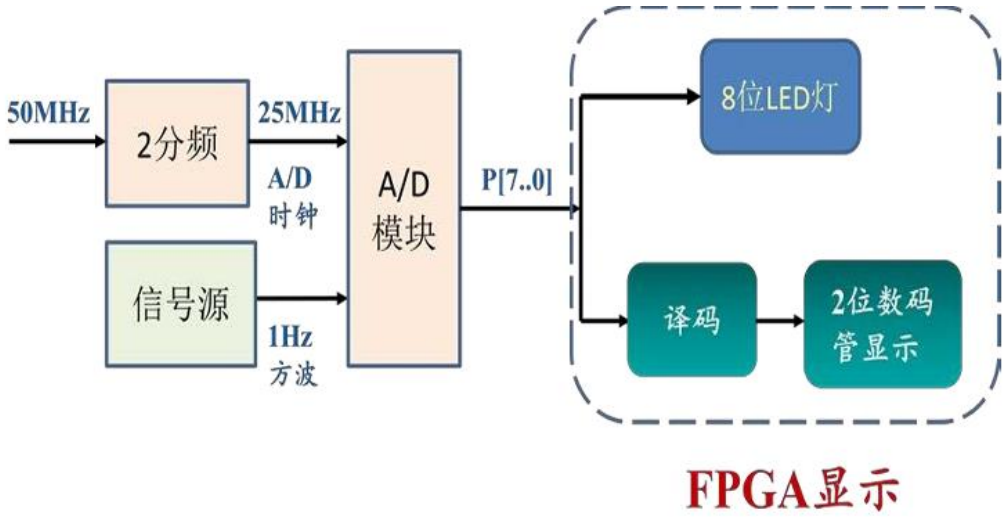
由上图可知，输入的模拟电压范围在-5V 至 5V 之间，按照正弦 波波形变化，最终得到的数据也是按照正弦波波形变化。AD9280 原理图：



2. 连接配置


(GPIO 0)
J4
[AB12] GPIO0_CLKIN0 1 2 GPIO0_D0 [AB16]
[AA12] GPIO0_CLKIN1 3 4 GPIO0_D1 [AA16]
[AA15] GPIO0_D2 5 6 GPIO0_D3 [AB15]
[AA14] GPIO0_D4 7 8 GPIO0_D5 [AB14]
[AB13] GPIO0_D6 9 10 GPIO0_D7 [AA13]
5V 11 12 GND
[AB10] GPIO0_D8 13 14 GPIO0_D9 [AA10]
[AB8] GPIO0_D10 15 16 GPIO0_D11 [AA8]
[AB5] GPIO0_D12 17 18 GPIO0_D13 [AA5]
[AB3] GPIO0_CLKOUT0 19 20 GPIO0_D14 [AB4]
[AA3] GPIO0_CLKOUT1 21 22 GPIO0_D15 [AA4]
[V14] GPIO0_D16 23 24 GPIO0_D17 [U14]
[Y13] GPIO0_D18 25 26 GPIO0_D19 [W13]
[U13] GPIO0_D20 27 28 GPIO0_D21 [V12]
3.3V 29 30 GND
[R10] GPIO0_D22 31 32 GPIO0_D23 [V11]
[Y10] GPIO0_D24 33 34 GPIO0_D25 [W10]
[T8] GPIO0_D26 35 36 GPIO0_D27 [V8]
[W7] GPIO0_D28 37 38 GPIO0_D29 [W6]
[V5] GPIO0_D30 39 40 GPIO0_D31 [U7]
VCC 5V
AD D0
AD D2
AD D4
AD D6
AD OTR
DA D7
DA D5
DA D3
DA D1
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
J2
GND
GND
AD D1
AD D3
AD D5
AD D7
AD CLK
DA CLK
DA D6
DA D4
DA D2
DA D0

3. 顶层框图


50MHz
2分频
25MHz
A/D
时钟
信号源
1Hz
方波
A/D
模块
P[7..0]
8位LED灯
译码
2位数码
管显示
FPGA显示

五、实验内容

1.分频

由于AD9280转换芯片支持的最大时钟频率为32Mhz，而FPGA 的系统时钟频率为50Mhz，所以需要先对时钟进行分频，将分频后的时钟作为AD9280转换芯片的驱动时钟。

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  ENTITY sdsy7_1hz IS
4  PORT(clk:IN STD_LOGIC;
5  clk_out1:OUT STD_LOGIC;
6  clk_out2:OUT STD_LOGIC);
7  END sdsy7_1hz;
8  ARCHITECTURE fwm OF sdsy7_1hz IS
9  CONSTANT m:INTEGER:= 2500000;
10 CONSTANT n:INTEGER:= 25000000;
11 SIGNAL tmp:STD_LOGIC;
12 SIGNAL tnp:STD_LOGIC;
13 BEGIN
14 PROCESS(clk,tmp)
15 VARIABLE cout : INTEGER:=0;
16 BEGIN
17 IF clk'EVENT AND clk='1' THEN
18 cout:=cout+1; --计数器+1
19 IF cout<=m THEN tmp<='0'; --计数小于等于 2500000，输出 0
20 ELSIF cout<m*2 THEN tmp<='1'; --计数小于 5000000，输出 1
21 ELSE cout:=0;
22 END IF;
23 END IF;
24 END PROCESS;
25 clk_out1<=tmp;
26 PROCESS(clk,tnp)
27 VARIABLE cout : INTEGER:=0;
28 BEGIN
29 IF clk'EVENT AND clk='1' THEN
30 cout:=cout+1; --计数器+1
31 IF cout<=n THEN tnp<='0'; --计数小于等于 25000000，输出 0
32 ELSIF cout<n*2 THEN tnp<='1'; --计数小于 50000000，输出 1
33 ELSE cout:=0;
34 END IF;
```

```
25 clk_out1<=tmp;
26 PROCESS(clk,tnp)
27 VARIABLE cout : INTEGER:=0;
28 BEGIN
29 IF clk'EVENT AND clk='1' THEN
30 cout:=cout+1; --计数器+1
31 IF cout<=n THEN tnp<='0'; --计数小于等于 25000000，输出 0
32 ELSIF cout<n*2 THEN tnp<='1'; --计数小于 50000000，输出 1
33 ELSE cout:=0;
34 END IF;
35 END IF;
36 END PROCESS;
37 clk_out2<=tnp;
38 END fwm;
```

2. A/D 模块将模拟信号转换成数字信号，记录数据如下。

频率 1Hz 方波信号	模拟信号	数字信号输出 8 位 LED[7..0]
		实测值
幅度+5 至-5 变化	电压为+5V 时	11111111（255）
	电压为-5V 时	00000000（000）
幅度+4 至-4 变化	电压为+4V 时	11100111（232）
	电压为-4V 时	00010111（024）
幅度+3 至-3 变化	电压为+3V 时	11001100（205）
	电压为-3V 时	00110000（049）
幅度+2 至-2 变化	电压为+2V 时	10110010（178）
	电压为-2V 时	01001010（074）
幅度+1 至-1 变化	电压为+1V 时	10100001（154）
	电压为-1V 时	01100100（102）

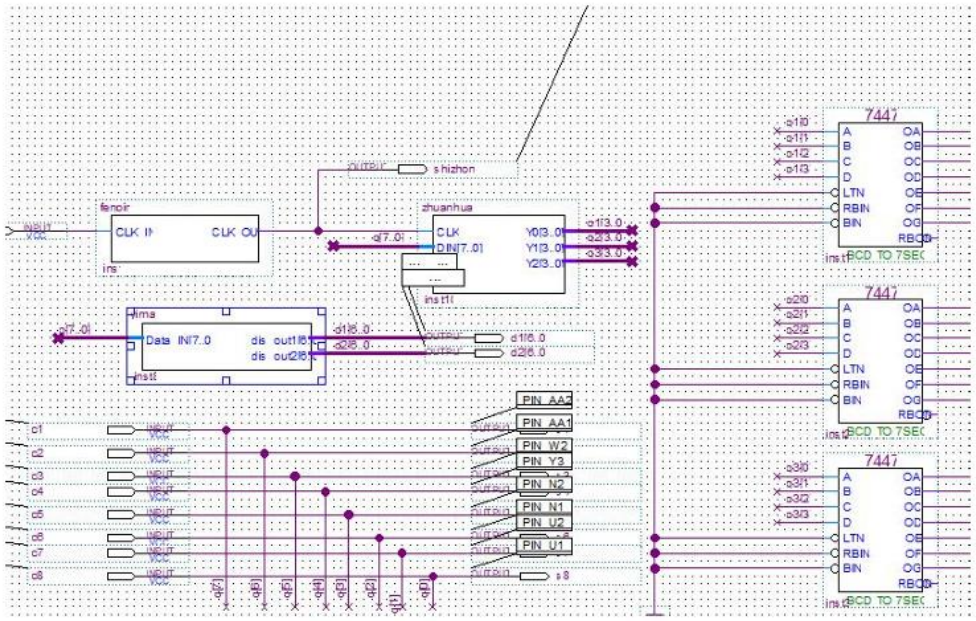
3.译码电路显示学号（“11111111”时显示“00”，“00000000”时显示“00”）

```
library ieee ;
use ieee.std_logic_1164.all;
ENTITY AD3 IS
port ( Data_IN  : in  std_logic_vector (7 downto 0) ;
           dis_out1 : out std_logic_vector (6 downto 0) ;
           dis_out2 : out std_logic_vector (6 downto 0));

end AD3;
ARCHITECTURE fwn OF AD3 IS
BEGIN
PROCESS(Data_IN)
BEGIN
CASE Data_IN IS
    WHEN"11111111"=>dis_out1<="0000000";dis_out2<="0010010";
    WHEN"00000000"=>dis_out1<="0000000";dis_out2<="0011001";
    WHEN others=>dis_out1<="1111111";dis_out2<="1111111";
END CASE;
END PROCESS;
END fwn;
```

4. 为了使得显示更加直观，可以将8位二进制数据，在三个七段数码管上显示。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity AD2 is
port(clk:in std_logic;
din:in std_logic_vector(7 downto 0);
y0,y1,y2:out std_logic_vector(3 downto 0));
end AD2;
architecture behav of AD2 is
type state is (s0,s1,s2);
signal present_state:state;
signal mid_in:std_logic_vector(7 downto 0);
signal d0,d1,d2:std_logic_vector(3 downto 0);
begin
process(clk)is
begin
if clk'event and clk='1' then
mid_in<=din;
present_state<=s0;
case (present_state) is
when s0=>
d0<="0000";d1<="0000";d2<="0000";
present_state<=s1;
when s1=>
if mid_in>="01100100" then mid_in<=mid_in-"01100100";
d2<=d2+1;
present_state<=s1;
elsif mid_in>="00001010" then mid_in<=mid_in-"00001010";
d1<=d1+1;
present_state<=s1;
elsif mid_in>="00000001" then
mid_in<=mid_in-"00000001";
d0<=d0+1;
present_state<=s1;
else
present_state<=s2;
end if;
when s2=>
y0<=d0;y1<=d1;y2<=d2;
present_state<=s0;
when others=>
present_state<=s0;
end case;
end if;
end process;
end behav;
```

5. 完整的原理图



六、实验过程中的问题

1. 电源和地线很重要，需要注意，不然无法得到正确的数据。

2. AD 模块需要频率信号，不然无法输出。

七、心得体会

1.通过实验对于模拟信号向数字信号的转变过程有了较好的认识。

2.实验项目是一个较为系统的过程，需要较多器件和引线，有一定的难度。不过熟练使用模块化的设计将大大降低难度。