装

订

线

EDA 课程设计实验报告

学　　院　信息工程学院

专　　业　通信工程

学　　号

姓　　名

任课教师

2013 年 10 月 30 日

一、FPGA 简介

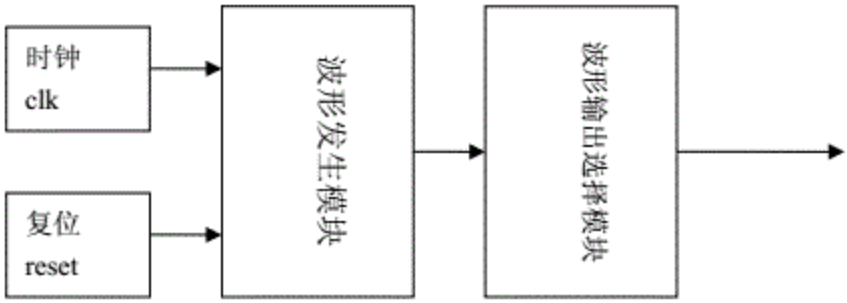
随着基于 FPGA 的 EDA 技术的发展和应用领域的扩大与深入，EDA 技术在电子信息、通信、自动控制及计算机等领域的重要性日益突出。作为一个学通信工程专业的学生，我们必须不断地去了解更多的新产品信息，这就更加要求我们对 EDA 有个全面的认识。信号发生器在我们的日常中有很重要的应用，用 VHDL 语言去实现设计将会使我们对本学科知识可以更好地掌握。

本设计是一个基于 VHDL 的采用自顶向下设计方法实现的信号发生器，该设计方法具有外围电路简单，程序修改灵活和调试容易等特点，并通过计算机仿真证明了设计的正确性。

二、题目分析

要求设计一个函数发生器，该函数发生器能够产生递增斜波、递减斜波、方波、三角波、正弦波、及阶梯波，并且可以通过选择开关选择相应的波形输出；系统具有复位的功能；通过按键确定输出的波形及确定是否输出波形。FPGA 是整个系统的核心，构成系统控制器，波形数据生成器，加法器，运算/译码等功能。

通过以上分析设计要求完成的功能，确定函数发生器可由递增斜波产生模块、递减斜波产生模块、三角波产生模块、阶梯波产生模块、正弦波产生模块、方波产生模块和输出波形选择模块组成，以及按键复位控制和时钟输入。由此可确定系统的总体原理框图为：



三、方案选择

1、波形函数发生方案对比选择

波形函数发生是本设计的最重要的部分，实现函数发生的途径也有很多，因此必须选择一种易于实现且精度高的方案，以此来提高本设计的实用性。

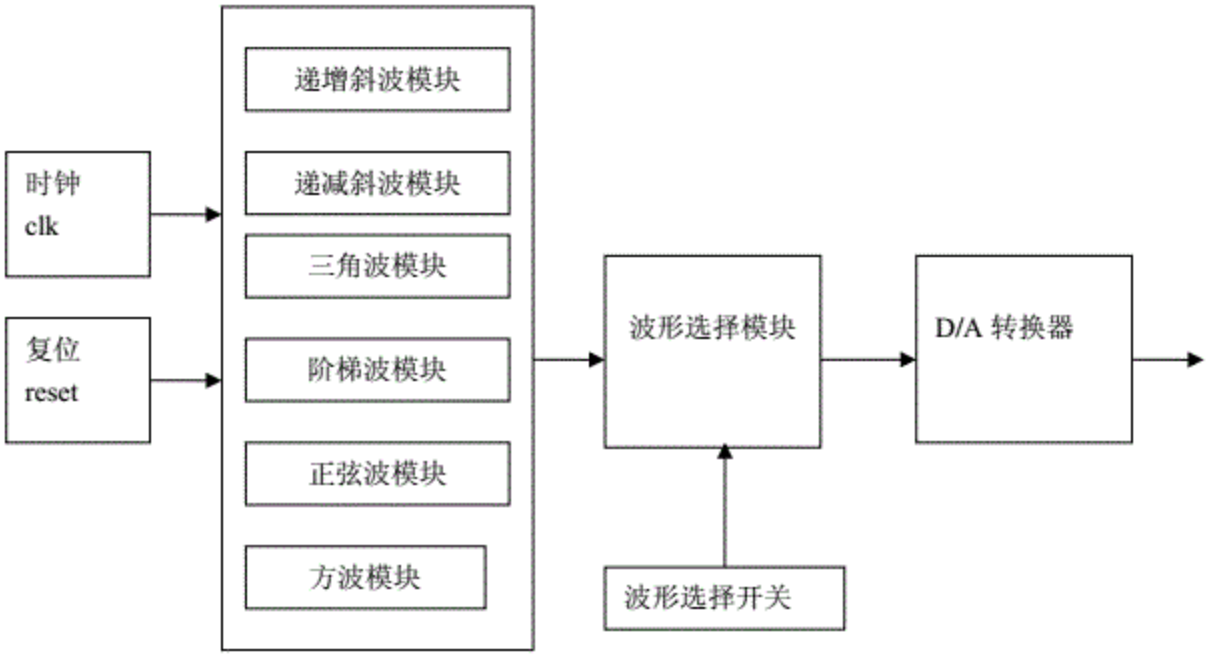
本信号发生器利用在系统编程技术和 FPGA 芯片产生。用 VHDL 语言编写程序，调试成功后下载至实验装置的芯片上，再利用外接 D/A 转换电路实现以上设计功能。此种方案完全可以生成设计要求的 6 种波形，而且通过软件仿真可以直观的观测的输出的波形参数，方便调试和更改波形参数，外围电路简单，减少器件损耗，精度高。

2、波形函数输出控制方式选择

利用 VHDL 语言写出数据选择器，然后每种函数发生器的输出和数据选择器输入相连接，通过控制开关选择对应的波形输出。方案二完全可以得到方案一的设计要求，而且只需一个 D/A 转换器就可以。电路不需要外部搭建，节约成本且控制简单方便。在实验课时候已经完成 8 选 1 数据选择器的设计制作，因此本次设计可以直接调用。此方案设计简便、节约制作元件和成本、控制简便等优点，因此作为波形函数输出控制方式。

四、系统细化框图

通过以上各个模块的分析最终确定函数信号发生器系统的最终整体的原理框图为：



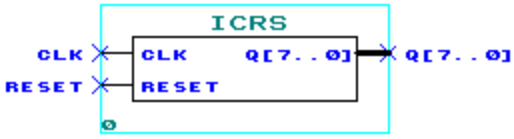
系统时钟输入后，通过复位开关选择是否产生波形，当各个模块产生相应的信号波形后，通过波形选择模块波形选择开关选泽输出不同的波形，再通过 D/A 转换器转换，就可以把数字信号（由 FPGA 输出）变成了相应模拟的信号波形。整个系统设计的核心就是 FPGA 部分。

五、各模块程序设计及仿真

根据自上而下的思路进行项目设计。明确每个模块的功能以后，开始编写各个模块的程序。

1、递增斜波模块

递增斜波 icrs 的 VHDL 程序如附录所示，其中 clk 是输入时钟端口，reset 为输入复位端口，q 为八位二进制输出端口。



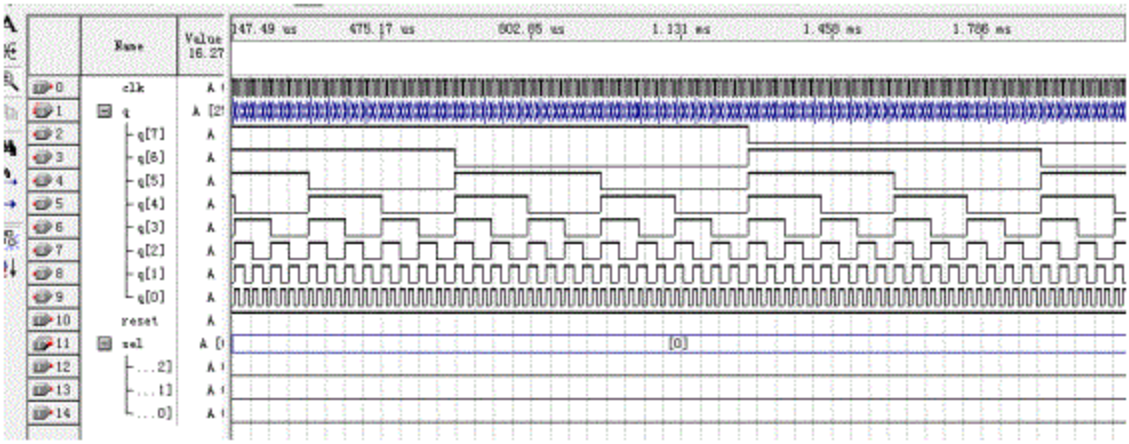


图 1 递增斜波模块仿真图

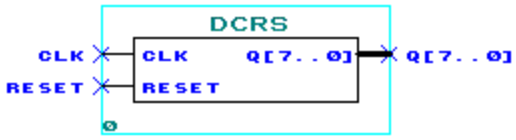
程序设计的当复位信号为 0 时,输出为 0，无对应的波形产生。当复位信号为 1 时，每当检测到时钟上升沿时，计数器值加 1，当增加到最大后清零。计数值增加呈现线性关系，因此输出的波形是递增的斜波。从仿真波形图也能看出这种变化规律。模块程序如下：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY icrs IS
   PORT(clk,reset: IN STD_LOGIC;
            q: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END icrs;
ARCHITECTURE behave OF icrs IS
BEGIN
PROCESS(clk,reset)
VARIABLE tmp : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
IF reset='0' THEN
        tmp:= "00000000";       --复位信号清零
ELSIF clk'EVENT AND clk='1' THEN
        IF tmp="11111111" THEN
      tmp:="00000000";          --递增到最大值清零
  ELSE
        tmp:=tmp+1;          --递增运算
      END IF;
```

```
  END IF;
    q<=tmp;
    END PROCESS;
END behave;
```

2、递减斜波模块

递减斜波 dcrs 的 VHDL 程序如附录所示，其中 clk 是输入时钟端口，reset 为输入复位端口，q 为八位二进制输出端口。



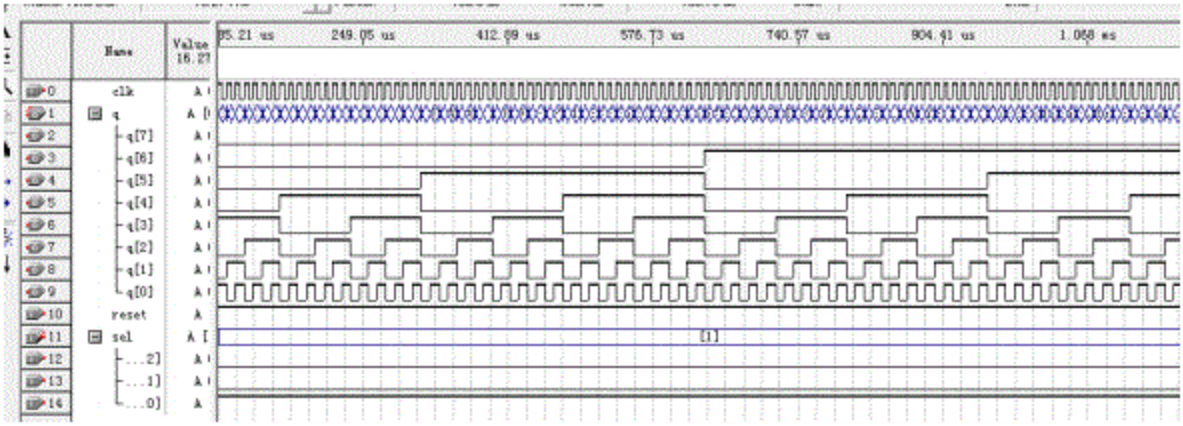


图 2 递减斜波模块仿真图

程序设计的是复位信号为 0 时输出为 0，无对应的波形产生。当复位信号为 1 时，当每当检测到时钟上升沿时，计数值减 1，当减到 0 后赋值到最大。计数值减少呈现线性关系，因此输出的波形是递减的斜波。从仿真波形图也能看出这种变化规律。模块程序如下：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY dcrs IS
PORT (clk,reset:IN STD_LOGIC;
        q:OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END dcrs;
ARCHITECTURE behave OF dcrs IS
BEGIN
PROCESS(clk,reset)
VARIABLE tmp:STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
IF reset='0' THEN
 tmp:="11111111";        --复位信号置最大值
```

```
ELSIF clk'EVENT AND clk='1' THEN          --检测时钟上升沿
IF tmp="00000000" THEN
 tmp:="11111111";        --递减到 0 置最大值
ELSE
 tmp:=tmp-1;          --递减运算
END IF;
END IF;
q<=tmp;
END PROCESS;
END behave;
```

3、三角波模块

三角波波 delat 的 VHDL 程序如附录所示，其中 clk 是输入时钟端口，reset 为输入复位端口，q 为八位二进制输出端口。

三角波波形是对称的，每边呈线形变化，所以可以根据数据做简单运算，

就可以得到三角波。

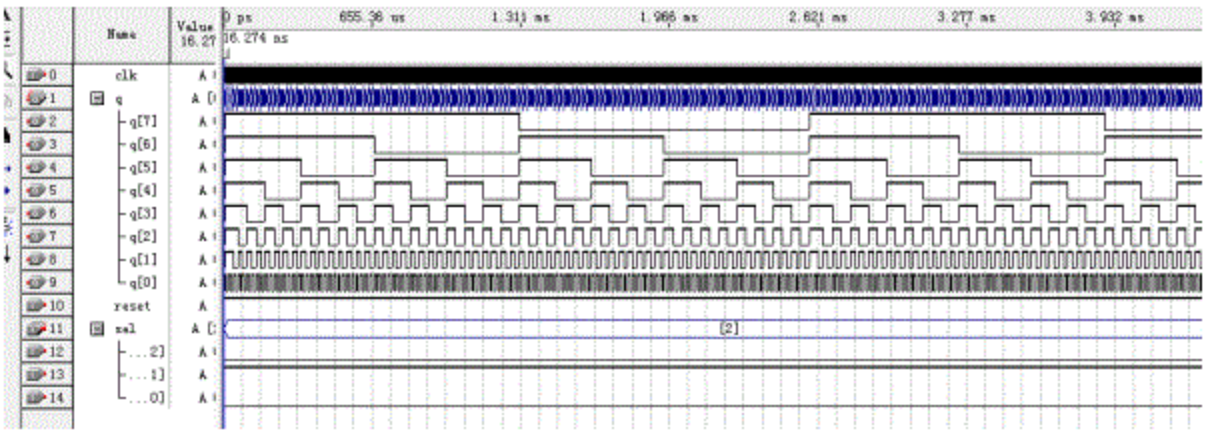


图 3 三角波模块仿真图

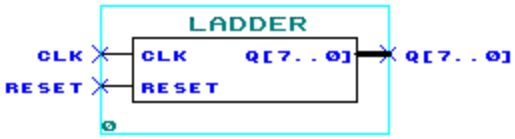
程序设计的是 reset 复位信号为 0 时输出为 0，无对应的波形产生。当复位信号为 1 时，当每当检测到时钟上升沿时，当计数的数据不是最大值时，数值做递增运算，当增大到最大时，然后再做递减运算，因此输出的波形便呈现出三角波的形状。从仿真波形图也能看出这种变化规律。模块程序如下：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY delta IS
        PORT(clk,reset:IN STD_LOGIC;
          q:OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
```

```
end delta;
ARCHITECTURE behave OF delta IS
BEGIN
 PROCESS(clk,reset)
 VARIABLE tmp:STD_LOGIC_VECTOR(7 DOWNTO 0);
  VARIABLE a:STD_LOGIC;
BEGIN
IF reset='0' THEN
    tmp:="00000000";          --复位信号为 0，置最小值
     ELSIF clk'EVENT AND clk='1' THEN          --检测时钟上升沿
 IF a='0' THEN
 IF tmp="11111110" THEN
tmp:="11111111";          --置最大值
a:='1';
 ELSE                    --不是最大值时递增
tmp:=tmp+1;           --递增运算
 END IF;
 ELSE
 IF tmp ="00000001" THEN
tmp:="00000000";          --置最小值
a:='0';
 ELSE             --a 为 1 时，执行递减运算
tmp:=tmp-1;       --递减运算
 END IF;
 END IF;
END IF;
q<=tmp;
END PROCESS;
END behave;
```

4、阶梯波模块

阶梯波 ladder 的 VHDL 程序如附录所示，其中 clk 是输入时钟端口，reset 为输入复位端口，q 为八位二进制输出端口。



阶梯波设计的是数据的递增是以一定的阶梯常数向上增加，所以输出的波形呈现是成阶梯状的，而不是，完全呈现是直线增长。模块程序如下：

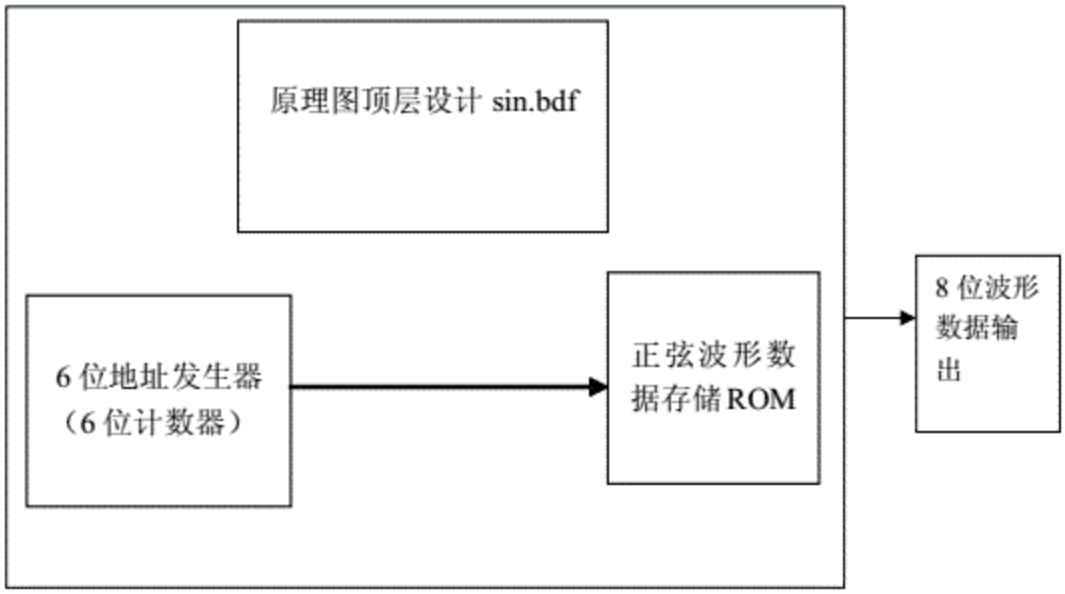
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ladder IS
```

```
PORT(clk,reset:IN STD_LOGIC;
  q:OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END ladder;
ARCHITECTURE behave OF ladder IS
BEGIN
PROCESS(clk,reset)
VARIABLE tmp: STD_LOGIC_VECTOR(7 DOWNTO 0); --定义内部变量
VARIABLE a: STD_LOGIC;
BEGIN
IF reset='0' THEN
  tmp:="00000000";        --复位信号为 0，置最小值
ELSIF clk'EVENT AND clk='1' THEN          --检测时钟上升沿
  IF a='0' THEN       --判断 a 数值，计数。
       IF tmp="11111111" THEN
tmp:="00000000";          --计数到最大清零
a:='1';
  ELSE
tmp:=tmp+16;          --阶梯常数为 16，可修改
a:='1';
  END IF;
  ELSE
a:='0';                   --循环计数
    END IF;
END IF;
q<=tmp;
END PROCESS;
END behave;
```

5、正弦波模块

正弦波模块由三个部分组成：6 位地址发生器、正弦信号数据 ROM 和原理图顶层设计文件。

结构图如下图所示：



上图所示的信号发生结构中图中，顶层文件 sin.bdf 在 FPGA 中实现，包含两个部分：ROM 的地址信号发生器，由 6 位计数器担任；一个正弦数据 ROM，由 LPM_ROM 模块构成，6 位地址线，8 位数据线，一个周期含有 64 个 8 位数据。LPM_ROM 底层是 FPGA 中的 EAB、ESB 或 M4K 等模块。地址发生器的时钟 CLK 的输入频率 F0 与每周期的波形数据点数以及 D/A 输出频率 F 的关系是：F=F0/64。

正弦波产生原理：通过循环不断地从波形数据 ROM 文件中依次读取正弦波一个周期在时域上 64 个采样点的波形数据送入波形 DAC，从而产生正弦波。正弦波的频率取决于读取数据的速度。

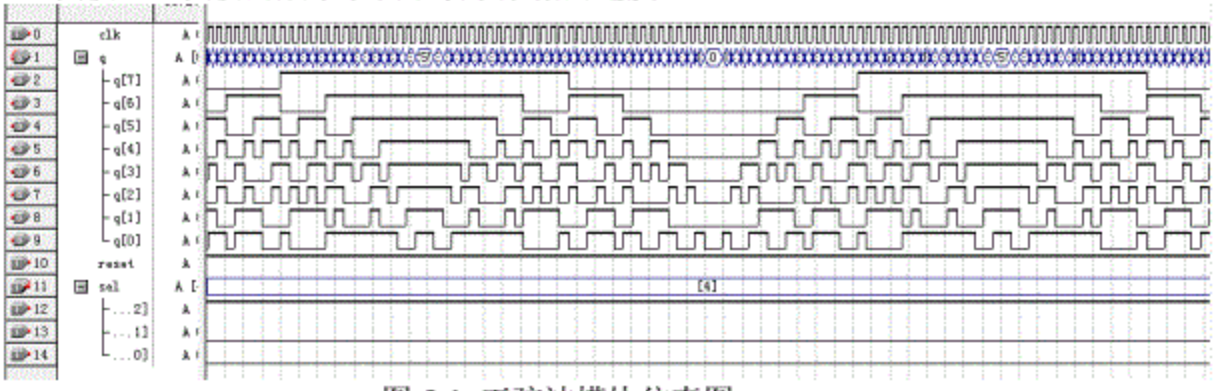


图 5-1 正弦波模块仿真图

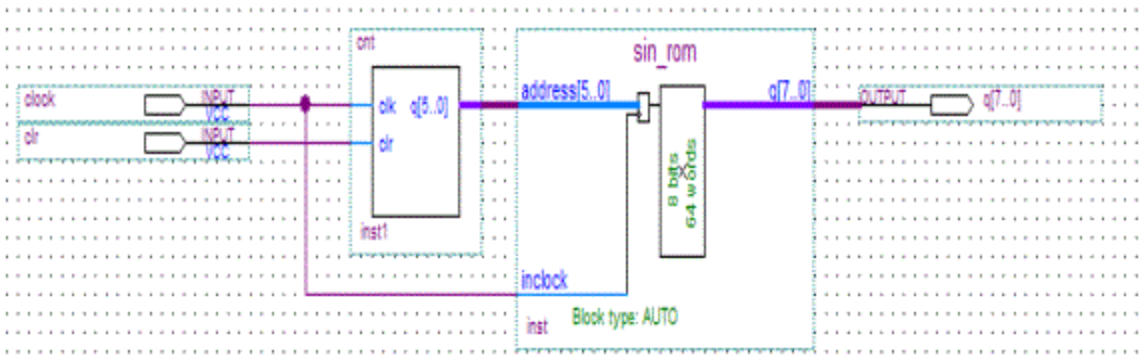


图 5-2 顶层文件原理图 sin.bdf

波形数据 ROM 文件 sin_rom.vhd 如下：

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.all;          --使用宏功能库中的所有元件
ENTITY sin_rom IS
    PORT
    (
        address     : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        inclock     : IN STD_LOGIC ;
        q           : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END sin_rom;
ARCHITECTURE SYN OF sin_rom IS
    SIGNAL sub_wire0   : STD_LOGIC_VECTOR (7 DOWNTO 0);
    COMPONENT altsyncram          --例化 altsyncram 元件，调用了 LPM 模块
altsyncram
    GENERIC (                                          --参数传递语句
        address_aclr_a          : STRING;
        init_file       : STRING;
        intended_device_family          : STRING;       --类属参量数据类型定义
        lpm_hint            : STRING;
        lpm_type            : STRING;
        numwords_a      : NATURAL;
        operation_mode          : STRING;
        outdata_aclr_a          : STRING;
        outdata_reg_a           : STRING;
        widthad_a           : NATURAL;
        width_a       : NATURAL;
        width_byteena_a       : NATURAL
    );
    PORT (
            clock0   : IN STD_LOGIC ;                  ---altsyncram 元件接口声明
            address_a     : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
            q_a : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
    END COMPONENT;
BEGIN
    q       <= sub_wire0(7 DOWNTO 0);
    altsyncram_component : altsyncram
    GENERIC MAP (
        address_aclr_a => "NONE",
```

```
        init_file => "sin_data.mif",
        intended_device_family => "Cyclone",          --参数传递映射
        lpm_hint => "ENABLE_RUNTIME_MOD=NO",
        lpm_type => "altsyncram",
        numwords_a => 64,                              --数据数量 64
        operation_mode => "ROM",                       --LPM 模式 ROM
        outdata_aclr_a => "NONE",                      --无异步地址清零
        outdata_reg_a => "UNREGISTERED",               --输出无锁存
        widthad_a => 6,                                --地址线宽度 6
        width_a => 8,                                  --数据线宽度 8
        width_byteena_a => 1
    )
    PORT MAP (
        clock0 => inclock,
        address_a => address,
        q_a => sub_wire0
    );
END SYN;
```

6 位地址信号发生器 cnt.vhd 如下：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity cnt is                                  --定义计数器的实体
    port(clk: in std_logic;
         clr: in std_logic;
         q: out std_logic_vector(5 downto 0));          --6 位输出地址线
end cnt;
architecture bhv of cnt is
begin
 process(clk,clr)
variable   cqi:std_logic_vector(5 downto 0);--定义内部变量
 begin
    if clr='0' then cqi:=(others =>'0');        --计数器异步复位
     elsif clk 'event and clk='1' then          --检测时钟上升沿
        cqi:=cqi+1;                             --计数
    end if;
q <=cqi;                --赋值，输出
 end process ;
 end bhv;
END SYN;
```

6、方波模块

方波模块的 square 的 VHDL 程序描述如下：其中 clk 为输入时钟端口，clr

为输入复位端口，q 为整数输出端口。

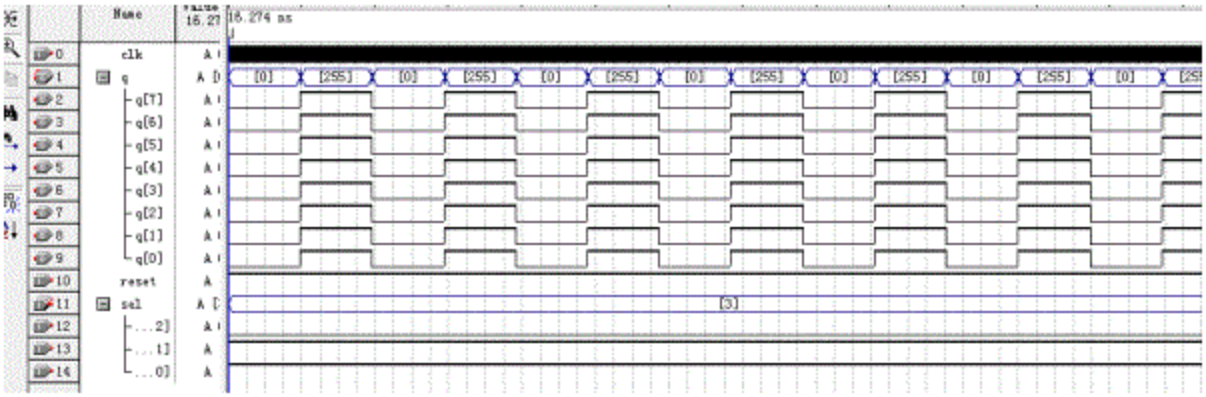


图 6 方波模块仿真图

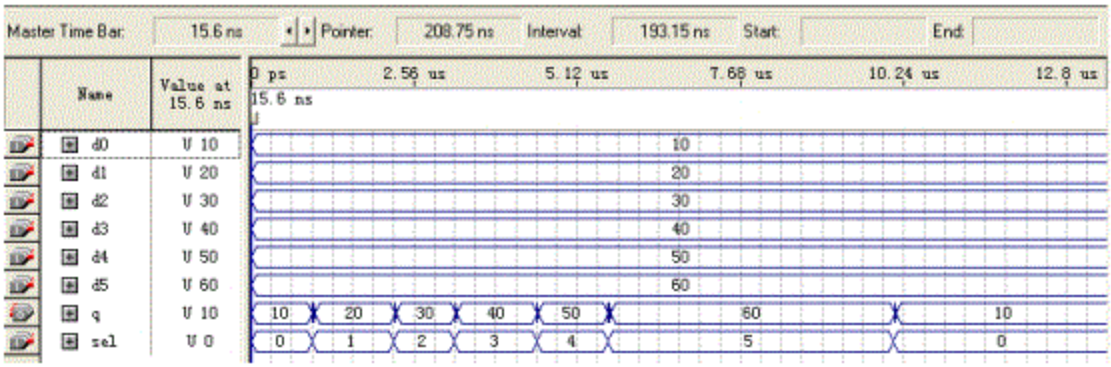
方波模块的设计是当内部计数 cnt 达到 64 时，根据输出标志 a 的数值输出对应的数值，当 a=0 输出 0，也即是方波周期中的低电平，当 a=1，输出 255，也即是方波周期中的高电平。连续的输出便成了观测到的方波波形。模块程序如下：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY square IS
PORT(clk,clr:IN STD_LOGIC;
q:OUT INTEGER RANGE 0 TO 255);
END square;
ARCHITECTURE behave OF square IS
SIGNAL a:BIT;
BEGIN
PROCESS(clk,clr)
VARIABLE cnt:INTEGER;          --定义内部整数变量
BEGIN
IF clr='0' THEN
  a<='0';
ELSIF clk'EVENT AND clk='1' THEN                --检测时钟上升沿
IF cnt<63 THEN       --计数 64 个点
cnt:=cnt+1;          --计数
  ELSE
cnt:=0;              --当计数的值大于 64 时，清零。
a<=NOT a;             --对内部 a 变量取反，a 变化已启动进程 END PROCESS;
  END IF;
END IF;
END PROCESS;
PROCESS(clk,a)
BEGIN
IF clk'EVENT AND clk='1' THEN
  IF a='1' THEN
q<=255; --a=1,         --输出一个波形周期内的高电平
```

```
 ELSE
q<=0; --a=0,            --输出一个波形周期的低电平。
 END IF;
END IF;
END PROCESS;
END behave;
```

7、输出波形选择模块

波形选择模块是一个设计位 6 选 1 的数据选择器，其中 sel 为波形数据选择端口，d0～d5 为 8 位二进制输入端口，q 为 8 位二进制输出端口。该模块可以根据外部开关的状态选择相应的波形输出。

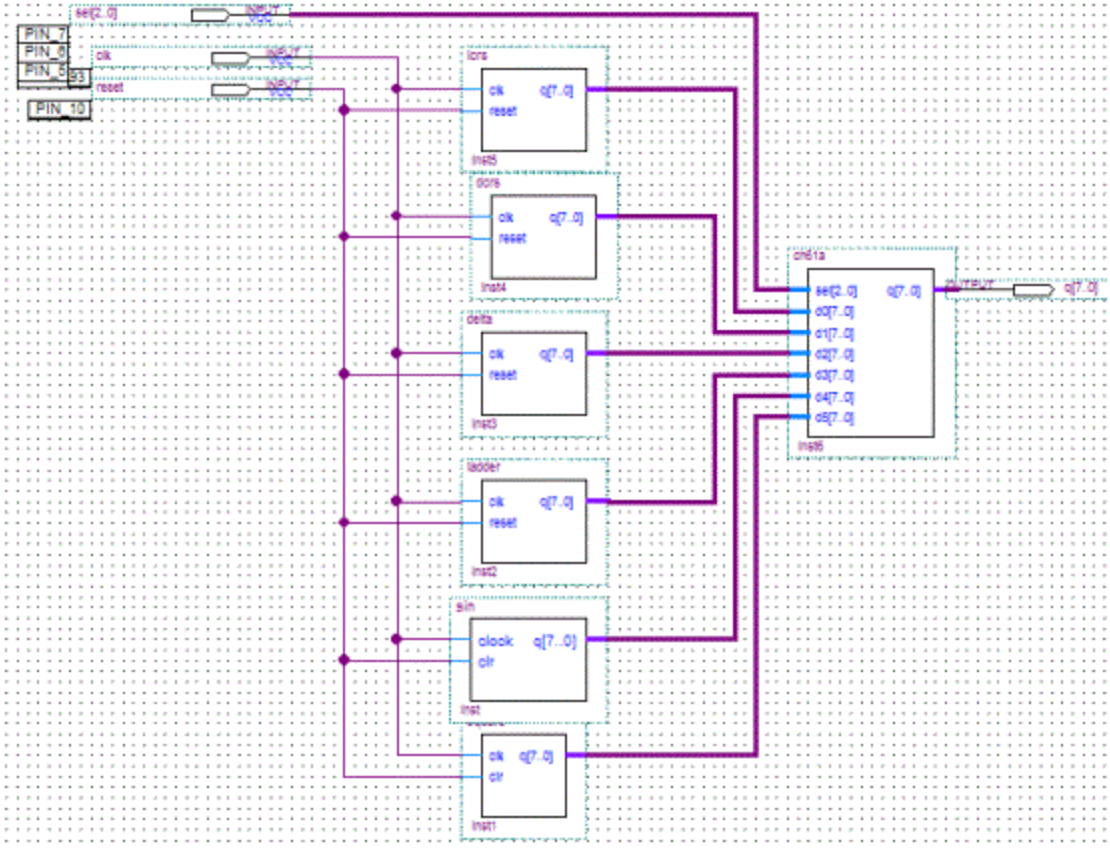


其选择 VHDL 程序如下：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ch61a IS
PORT(sel:IN STD_LOGIC_VECTOR(2 DOWNTO 0);
d0,d1,d2,d3,d4,d5:IN STD_LOGIC_VECTOR(7 DOWNTO 0);
q:OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END ch61a;
ARCHITECTURE behave OF ch61a IS
BEGIN
PROCESS(sel)
BEGIN
CASE sel IS
WHEN"000"=>q<=d0;       --递增波形输出
WHEN"001"=>q<=d1;       --递减波形输出
WHEN"010"=>q<=d2;       --三角波形输出
WHEN"011"=>q<=d3;       --阶梯波形输出
WHEN"100"=>q<=d4;       --正弦波形输出
WHEN"101"=>q<=d5;       --方波输出
WHEN OTHERS=>NULL;
END CASE;
END PROCESS;
END behave;
WHEN OTHERS=>NULL;
```

六、系统联调测试分析

通过以上各个模块的细化和分析，最终在 Quartus Ⅱ中完成了整个系统的联合调试，并通过嵌入式逻辑分析的方法回读输出信号的波形符合设计的要求。调试整个系统了原理图如下图所示：



1. 调试的结果如下：（复位信号 reset 高电平，低电平为不输出）

（1）第一次 sel 选择值设为 0，输出为递增波，从图中可以看出，输出的波形成线性递增，结果正确。

（2）第二次 sel 选择值设为 1，输出为递减波，从图中可以看出，输出的波形成线性递减，结果正确。

（3）第三次 sel 的值设为 2，输出为三角波，其仿真波形如下图所示，输出波形线性增大到最大后，再线性减小。

（4）第四次 sel 的值设为 3，其输出的波形是阶梯波，其仿真波形见下图，波形递增常数为 16，结果正确。

（5）第五次 sel 的值设为 4，其输出的波形是正弦波，从图中可以看出，输出的数据的变化规律是正弦规律。

（6）第六次 sel 的值设为 5，其输出的波形是方波，从图中仿真的结果可以看出，输出的波形变化规律是按方波规律周期性变化的。

（7）当设置为其他值时无波形输出

七、设计结论

本设计以函数信号发生器的功能为设计对象，运用 EDA 技术的设计方法，进行各种波形的输入设计、设计处理和器件编程。在 VHDL 语言的编写中按照语言描述规范，实现了几种波形的软件设计和具体逻辑元件结构的硬件映射。结合 FPGA 的开发集成环境 Quartus2 软件，产生了函数信号发生器的各种信号，同时完成了时序和功能仿真。实验表明采用该方法能准确的产生三角波、阶梯波、正弦波等设计产生的波形，实现了信号发生器的功能。

本设计的函数信号发生器在设计上由于设计时考虑的不够全面虽然完成了函数信号的产生，但不够完善。要做成完整实用的信号源还应考虑设计包含的功能有：

（1）用键盘输入编辑生成上述 6 种波形（同周期）的线性组合波形；
（2）具有波形存储功能；
（3）输出波形的频率范围可调，频率步进；
（4）输出波形幅度可调，步进调整；
（5）具有显示输出波形的类型、重复频率（周期）和幅度的功能；
（6）用键盘或其他输入装置产生任意波形；
（7）波形占空比可调等。

八、心得体会

一个学期的 EDA 学习，使我获益良多。在这期间学习了 EDA 的基本知识、常用的 EDA 的工具 Quartus2 的使用方法、对大规模可编程器件的结构和工作原理也有了一定的了解；掌握了原理图和 VHDL 输入的基本设计方法；对 VHDL 语言的语法结构、编程结构也都有了一定的掌握；结合实验课学会了编程下载和硬件测试等内容；对 Quartus2 软件的嵌入式逻辑分析仪的使用和宏功能模块的调用也掌握了一些基本的操作；配合着实验课初步学会了自顶向下的设计方法，明白了如何用这种方法去实现一个系统的设计。但这些内容掌握的程度还不深入，要想能够融会贯通必须用更多的时间去深入学习。

这些内容的学习，增强了我对 EDA 设计的兴趣，具备了这些基本知识，为今后的自主学习奠定了良好的基础。