实验一：QUARTUS II 软件使用及

组合电路设计仿真

实验目的：

学习 QUARTUS II软件的使用，掌握软件工程的建立，VHDL源文件的设计和波形仿真等基本内容。

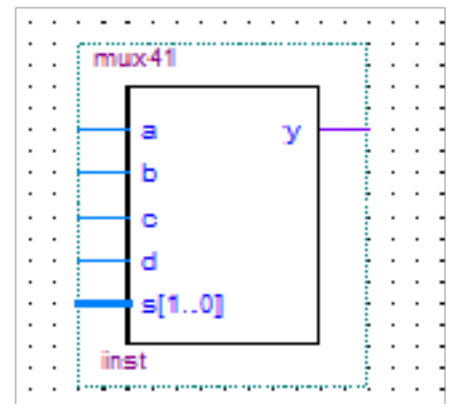
实验内容：

1. 四选一多路选择器的设计

基本功能及原理 ：

选择器常用于信号的切换，四选一选择器常用于信号的切换，四选一选择器可以用于 4 路信号的切换。四选一选择器有四个输入端 a,b,c,d ，两个信号选择端 s(0) 和 s(1) 及一个信号输出端 y。当 s 输入不同的选择信号时，就可以使 a,b,c,d 中某一个相应的输入信号与输出 y 端接通。

逻辑符号如下：



程序设计：

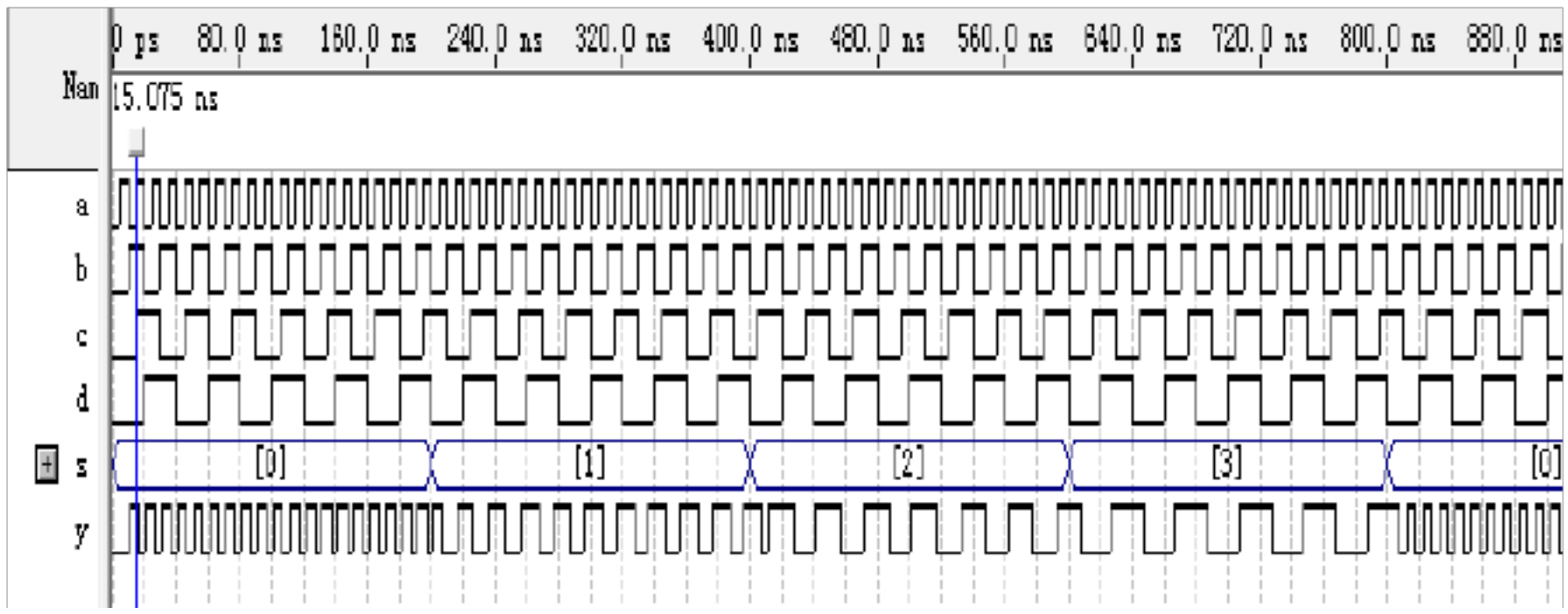
```
library ieee;
use ieee.std_logic_1164.all;
entity mux41 is
port
    (
      a,b,c,d: in  std_logic;
             s: in  std_logic_vector(1 downto 0);
             y: out std_logic
     );

end mux41;

architecture bhv of mux41 is
begin
  process(s)
    begin
     case s is
       when "00"=>y<=a;
       when "01"=>y<=b;
       when "10"=>y<=c;
       when "11"=>y<=d;
     end case;
  end process;
end bhv;
```

软件编译：

在编辑器中输入并保存了以上四选一选择器的 VHDL源程序后就可以对它进行编译了，编译的最终目的是为了生成可以进行仿真、定时分析及下载到可编程器件的相关文件。

仿真分析：

仿真结果如下图所示



分析：

由仿真图可以得到以下结论：

当 s=0(00) 时 y=a; 当 s=1(01) 时 y=b；当 s=2(10) 时 y=c；当 s=3(11) 时 y=d。符合我们最开始设想的功能设计，这说明源程序正确。

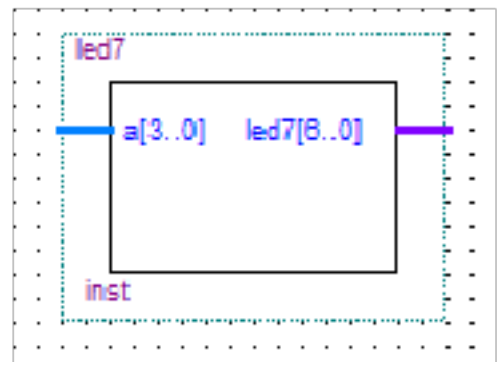
2. 七段译码器程序设计

基本功能及原理：

七段译码器是用来显示数字的，7 段数码是纯组合电路，通常的小规模专用 IC，如 74 或 4000 系列的器件只能作十进制 BCD 码译码，然而数字系统中的数据处理和运算都是 2 进制的，所以输出表达都是 16 进制的，为了满足 16 进制数的译码显示，最方便的方法就是利用 VHDL 译码程序在 FPGA 或 CPLD 中实现。本项实验很容易实现这一目的。输出信号的 7 位分别接到数码管的 7 个段，本实验中用的数码管为共阳极的，接有低电平的段发亮。

数码管的图形如下

七段译码器的逻辑符号：



程序设计：

```
library ieee;
use ieee.std_logic_1164.all;
entity led7 is
    port
    (a   :in  std_logic_vector(3downto 0);

     led7: out std_logic_vector(6 downto 0));
end ;
architecture bhv of led7 is
begin
   process(a)
   begin
       case a(3downto 0) is
         when "0000"=> led7 <="1000000";
         when "0001"=> led7 <="1111001";
         when "0010"=> led7 <="0100100";
         when "0011"=> led7 <="0110000";
         when "0100"=> led7 <="0011001";
         when "0101"=> led7 <="0010010";
         when "0110"=> led7 <="0000010";
         when "0111"=> led7 <="1111000";
         when "1000"=> led7 <="0000000";
         when "1001"=> led7 <="0010000";
         when "1010"=> led7 <="0001000";
         when "1011"=> led7 <="0000011";
         when "1100"=> led7 <="1000110";
         when "1101"=> led7 <="0100001";
         when "1110"=> led7 <="0000110";
         when "1111"=> led7 <="0001110";
         when others =>null;
       end case;
  end process;
end bhv;
```

软件编译：

在编辑器中输入并保存了以上七段译码器的 VHDL源程序后就可以对它进行编译了，编译的最终目的是为了生成可以进行仿真、定时分析及下载到可编程器件的相关文件

。

仿真分析：

仿真结果如下图所示：

分析：

由仿真的结果可以得到以下结论：

当 a=0(0000)时 led7=1000000 此时数码管显示 0；

当 a=1(0001)时 led7=1111001 此时数码管显示 1；

当 a=2(0010)时 led7=0100100 此时数码管显示 2；

当 a=3(0011)时 led7=0110000 此时数码管显示 3；

当 a=4(0100)时 led7=0011001 此时数码管显示 4；

当 a=5(0101)时 led7=0010010 此时数码管显示 5；

当 a=6(0110)时 led7=0000010 此时数码管显示 6；

当 a=7(0111)时 led7=1111000 此时数码管显示 7；

当 a=8(1000)时 led7=0000000 此时数码管显示 8；

当 a=9(1001)时 led7=0010000 此时数码管显示 9；

当 a=10(1010)时 led7=0001000 此时数码管显示 A；

当 a=11(1011)时 led7=0000011 此时数码管显示 B；

当 a=12(1100)时 led7=1000110 此时数码管显示 C；

当 a=13(1101)时 led7=0100001 此时数码管显示 D；

当 a=14(1110)时 led7=0000110 此时数码管显示 E；

当 a=15(1111)时 led7=0001110 此时数码管显示 F；

这完全符合我们最开始的功能设计，所以可以说明源 VHDL 程序是正确的。

实验心得：

通过这次实验，我基本掌握了 QUARTUS II 软件的使用，也掌握了软件工程的建立，VHDL源文件的设计和波形仿真等基本内容。在实验中，我发现 EDA这门课十分有趣，从一个器件的功能设计到程序设计，再到编译成功，最后得到仿真的结果，这其中的每一步都需要认真分析，一遍又一遍的编译，修改。当然，中间出现过错误，但我依然不放弃，一点一点的修改，验证，最终终于出现了正确的仿真结果，虽然有一些毛刺，但是总的来说，不影响整体的结果。

实验二：计数器设计与显示

实验目的：

（1）熟悉利用 QUARTUS II 中的原理图输入法设计组合电路，掌握层次化的设计方法；

（2）学习计数器设计，多层次设计方法和总线数据输入方式的仿真，并进行电路板下载演示验证。

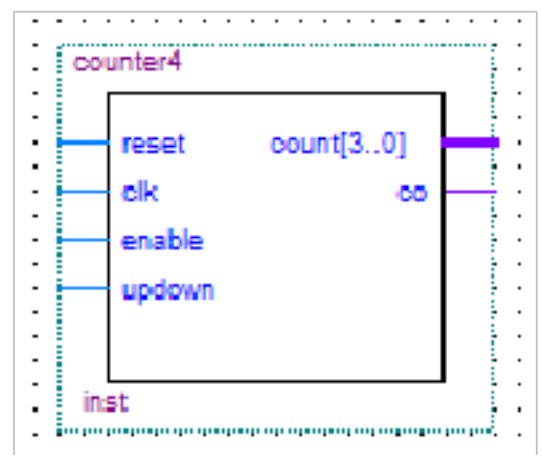
实验内容：

1. 完成计数器设计

基本功能及原理：

本实验要设计一个含有异步清零和计数使能的 4 位二进制加减可控计数器，即有一个清零端和使能端，当清零端为 1 时异步清零，即所有输出值都为 0，当使能端为 0 时，计数器停止工作，当使能端为 1 时，正常工作，由时钟控制。另外，还应该有一个控制端，当控制端为 0 时，进行减法运算，当控制端为 1 时，进行加法运算。输出端有输出值和进位端，当进行加法运算时，输出值递增，当减法运算时，输出值递减，同时进位端进行相应的变化。

4 位二进制加减计数器的逻辑符号：



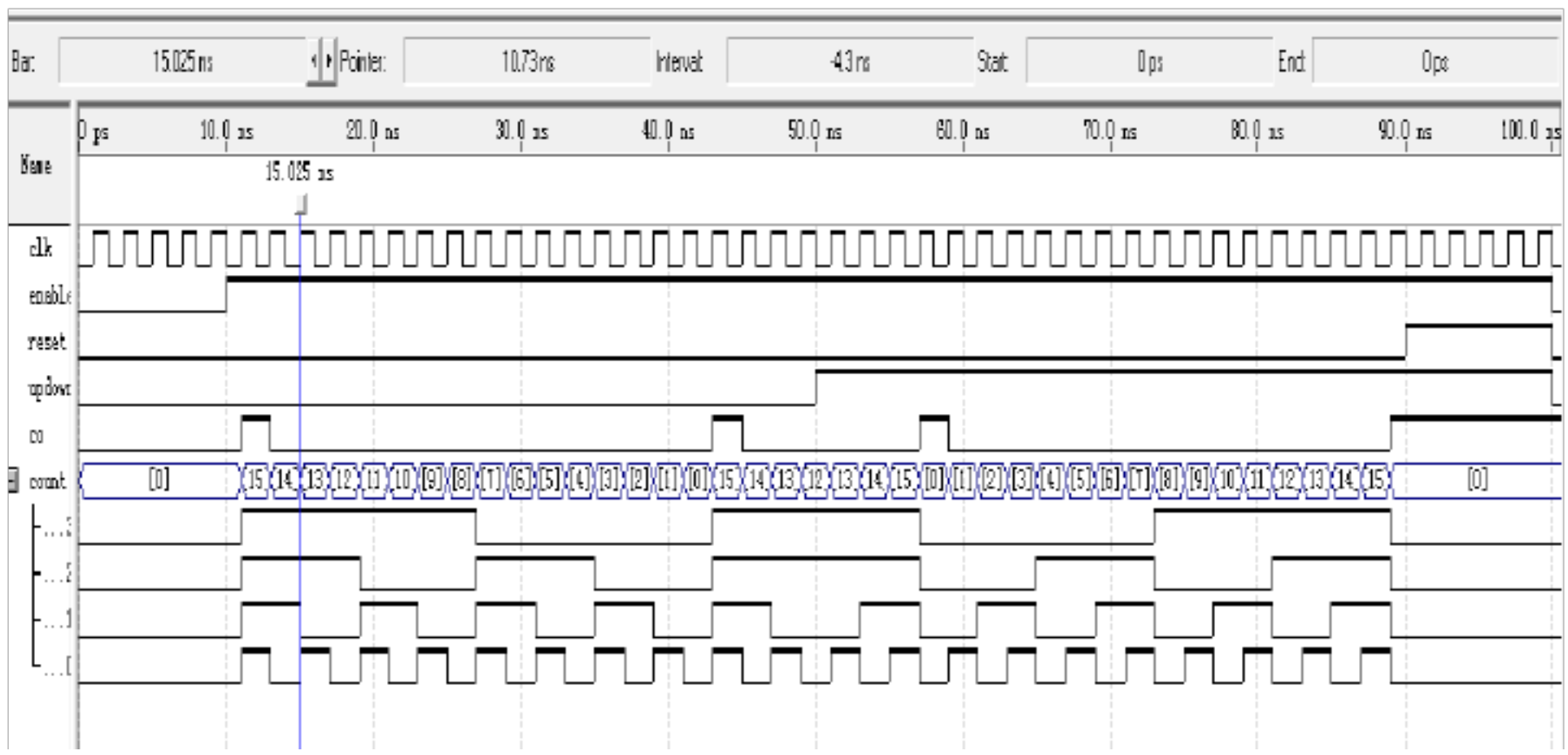
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter4 is

    port
    (

        reset: in std_logic;
        clk: in std_logic;
        enable: in std_logic;
        updown: in std_logic;
        count: out std_logic_vector(3 downto 0);
        co: out std_logic
    );
end counter4;
architecture arc of counter4 is
  signal cnt:std_logic_vector(3 downto 0);
begin
  process(clk,reset)
  begin
      if reset='1' then
            cnt<=(others=>'0');
      elsif clk'event and clk='1' then
          if enable='1' then
           if updown='1' then
             if cnt="1111"then
             co<='1';
             cnt<="0000";
             else
             co<='0';
             cnt<=cnt+1;
             end if;
           else
             if cnt="0000" then
             co<='1';
             cnt<="1111";
             else
             co<='0';
             cnt<=cnt-1;
             end if;
           end if;
          end if;
       end if;
end process;
count<=cnt;
end arc;
```

软件编译：

在编辑器中输入并保存了以上 4 位二进制加减计数器的 VHDL 源程序后就可以对它进行编译了，编译的最终目的是为了生成可以进行仿真、定时分析及下载到可编程器件的相关文件。

仿真分析：

仿真结果如下：



分析：

由仿真图可以得到以下结论：

当 enable 端为 0 时，所有数值都为 0，当 enable 端为 1 时，计数器正常工作；当 reset 端为 1 时，异步清零，所有输出数值为 0，当 reset 端为 0 时，正常工作；当 updown 端为 0 时，进行减法运算，当 updown 为 1 时，进行加法运算；另外，当程序进行减法运算时，出现借位时，co 为 1，其余为 0，当进行加法运算时，出现进位时，co 为 1，其余

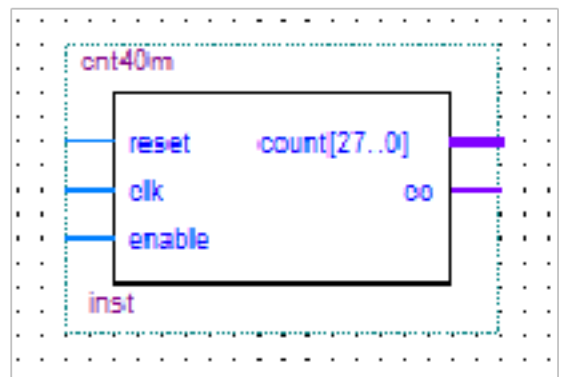
。图中所有的功能与我们设计的完全一样，所以说明源程序正确。

2.50M分频器的设计

基本功能及原理：

50M分频器的作用主要是控制后面的数码管显示的快慢。即一个模为 50M的计数器，由时钟控制，分频器所有的端口基本和上述 4 位二进制加减计数器的端口一样，原理也基本相同。分频器的进位端（co）用来控制加减计数器的时钟，将两个器件连接起来。

50M分频器的逻辑符号如下：



程序设计：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity cnt40m is
    port
    (   reset: in std_logic;
        clk: in std_logic;
        enable: in std_logic;
        count: out std_logic_vector(27 downto 0);
        co: out std_logic
    );
end cnt40m;
architecture bhv of cnt40m is
  signal cnt:std_logic_vector(27 downto 0);
begin

  process(clk,reset)
  begin
      if reset='1' then
            cnt<=(others=>'0');
      elsif clk'event and clk='1' then
          if enable='1' then
```