2016 年《综合课程设计 2-1》课程实验报告（第三类实验）

实验题三：设计 8086 逻辑地址到物理地址的转换电路

姓名：王科

学号：1310583

专业：计算机科学与技术

完成日期：2016/4/9

目录

一、 实验内容

本次实验是使用图形设计方法与VHDL编程相结合在FPGA芯片内构建独立逻辑实验，要求设计8086逻辑地址到物理地址的转换电路。逻辑地址的段地址和段内偏移量均由逻辑开关提供，转换成的 20 位物理地址不要求寄存，直接送LED显示。

二、 实验原理

1.逻辑地址与物理地址的转换关系

逻辑地址（Logical Address） 是指由程序产生的与段相关的偏移地址部分。其表达形式为“段地址：段内偏移地址”.例如，在进行C语言指针编程中，可以读取指针变量本身值(&操作)，实际上这个值就是逻辑地址，它是相对于当前进程数据段的地址，不和绝对物理地址相干。只有在Intel实模式下，逻辑地址才和物理地址相等（因为实模式没有分段或分页机制,Cpu不进行自动地址转换）；逻辑也就是在 Intel 保护模式下程序执行代码段限长内的偏移地址（假定代码段、数据段如果完全一样）。应用程序员仅需与逻辑地址打交道，而分段和分页机制对您来说是完全透明的，仅由系统编程人员涉及。应用程序员虽然自己可以直接操作内存，那也只能在操作系统给你分配的内存段操作。

物理地址（Physical Address） 是指出现在CPU外部地址总线上的寻址物理内存的地址信号，是地址变换的最终结果地址。CPU与存储器进行数据交换时在地址总线上提供的20位地址信息称为物理地址.如果启用了分页机制，那么线性地址会使用页目录和页表中的项变换成物理地址。如果没有启用分页机制，那么线性地址就直接成为物理地址了。

8086/8088中，每一个存储单元都有一个唯一的20位地址，称此地址为该存储单元的物理地址。CPU访问存储器时，必须先确定所要访问的存储单元地址才能取得该单元的内容。20位的物理地址由16位的段地址和16位的段内偏移地址计算得到。段地址是每一逻辑段的起始地址，必须是每个小节的首地址，其低4位一定是0，于是在保留段地址时，可以只取段地址的高16位。偏移地址则是在段内相对于段起始地址的偏移值。因此任一存储单元物理地址的计算方法如下：

物理地址=16×段地址+段内偏移地址

2. 设计概述

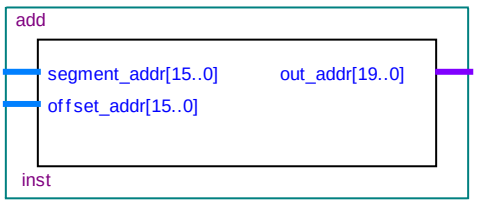
8086 逻辑地址到物理地址的转换电路，是将两个 16 位的逻辑地址，转换成一个 20 位的物理地址，其本质是将段地址乘以 16 加上偏移地址计算得到物理地址，其中，输入输出特性如下图：

2.1 输入输出特性表

INPUT		OUTPUT
段地址	偏移地址	物理地址
Segment_addr[15..0]	Offset_addr[15..0]	Out_addr[19..0]=Segment_addr[15..0]*16+ Offset_addr[15..0]

2.2 设计外部端口

按照设计要求，即需要 32 个输入端口和 20 个输出端口，其中输入端口包括 16 个段地址信号端口和 16 个偏移地址信号端口，输出端口是一个 20 位的转换过后的物理地址。其整体外部端口如下：



3. 设计内部实现逻辑

设计这次的 8086 逻辑地址到物理地址的转换电路其本质在于实现一个 20 位加法逻辑。先将段地址（以二进制形式表示）左移 4 位，形成一个 20 十位的二进制数，然后与偏移地址相加。在构建加法逻辑中，我使用了 LOOP 循环，即循环 20 次，从第零位开始每一次循环将当前一位的加数、被加数和上次计算产生的进位加起来，产生当前这一位的结果和一个进位。这样最终得到了一个 20 位的加法结果。这就是转换过后的物理地址。

4. 设计描述

使用自底向上设计方法，先使用 VHDL 语言设计一个加法器，为其生成元部件，然后设计的主体框架是一个使用 quartus 图形化设计方法设计的一个逻辑地址到物理地址的转换电路，并为其设置引脚并编译生成工程文件。

三、 实验步骤

1. 设计加法器的端口、内部 VHDL 逻辑程序

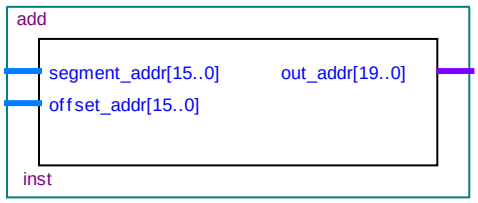
按照实验要求设计一个加法器的元部件，其输入端口包括两个 16 位的输入端口和一个 20 位的输出端口，vhdl 语言设计端口如下：

```
PORT(
      segment_addr: IN STD_LOGIC_VECTOR(15 downto 0);
      offset_addr:   IN STD_LOGIC_VECTOR(15 downto 0);
      out_addr:   OUT STD_LOGIC_VECTOR(19 downto 0)
 );
```

在其内部实现逻辑功能代码如下：

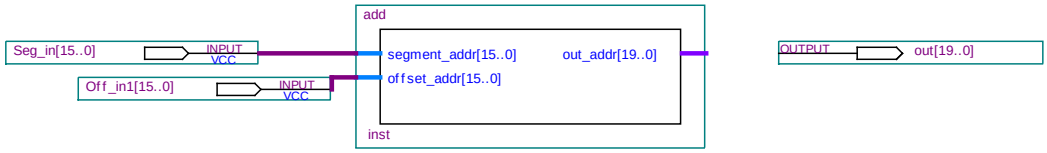
```
ARCHITECTURE add_architecture OF add IS
   signal a:std_logic_vector(19 downto 0);
   signal b:std_logic_vector(19 downto 0);
begin
   behavior:process(segment_addr,offset_addr) is
   variable carry_in:std_logic;
   variable carry_out:std_logic;
   variable op2:std_logic_vector(19 downto 0);
   begin
      a<= segment_addr&"0000";
      b<= "0000"&offset_addr;
      op2:=b;
      for index in 0 to 19 loop
      carry_in:=carry_out;
      out_addr(index)<=a(index) xor op2(index) xor carry_in;
      carry_out:=(a(index)and op2(index)) or (carry_in and (a(index) xor op2(index)));
   end loop;
   end process;
 end add_architecture;
```

编译完成后为其生成.bdf 元器件如下：



2.使用图形化方法设计原理图

使用 quartus 包含上述设计的元件，设计原理图如下：



3 观察 FPGA 电路板，将输入输出端口进行引脚绑定

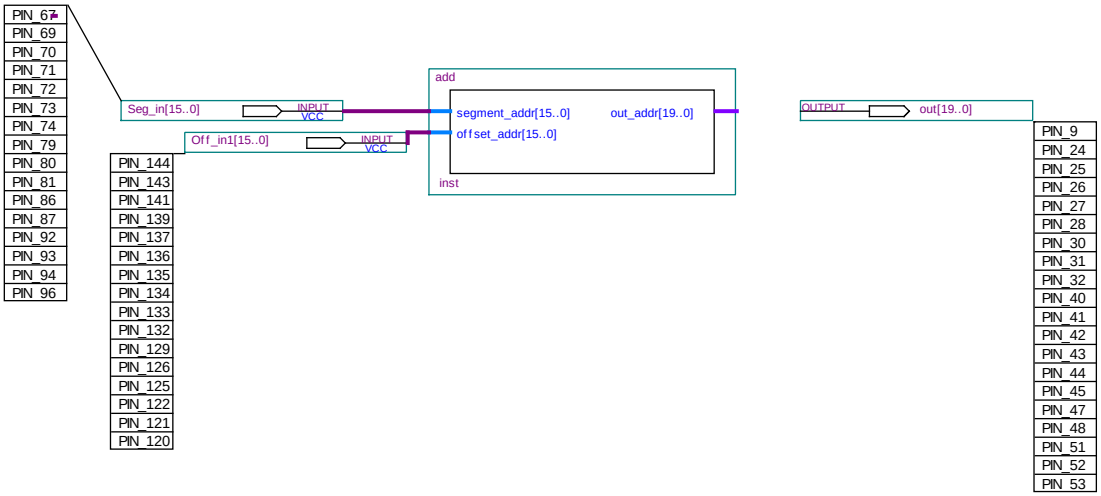
根据 FPGA 电路板上空闲的剩余引脚，在这次实验中个输入输出端对应的引脚号如下表：

	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard	Reserved	Group	Current Strength
1	Off_in1[15]	Input	PIN_144	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
2	Off_in1[14]	Input	PIN_143	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
3	Off_in1[13]	Input	PIN_141	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
4	Off_in1[12]	Input	PIN_139	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
5	Off_in1[11]	Input	PIN_137	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
6	Off_in1[10]	Input	PIN_136	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
7	Off_in1[9]	Input	PIN_135	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
8	Off_in1[8]	Input	PIN_134	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
9	Off_in1[7]	Input	PIN_133	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
10	Off_in1[6]	Input	PIN_132	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
11	Off_in1[5]	Input	PIN_129	2	B2_N1	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
12	Off_in1[4]	Input	PIN_126	2	B2_N0	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
13	Off_in1[3]	Input	PIN_125	2	B2_N0	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
14	Off_in1[2]	Input	PIN_122	2	B2_N0	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
15	Off_in1[1]	Input	PIN_121	2	B2_N0	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
16	Off_in1[0]	Input	PIN_120	2	B2_N0	3.3-V LVTTL (default)		Off_in1[15..0]	24mA (default)
17	out[19]	Output	PIN_9	1	B1_N0	3.3-V LVTTL (default)		out[19..0]	24mA (default)
18	out[18]	Output	PIN_24	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
19	out[17]	Output	PIN_25	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
20	out[16]	Output	PIN_26	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
21	out[15]	Output	PIN_27	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
22	out[14]	Output	PIN_28	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
23	out[13]	Output	PIN_30	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
24	out[12]	Output	PIN_31	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
25	out[11]	Output	PIN_32	1	B1_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
26	out[10]	Output	PIN_40	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
27	out[9]	Output	PIN_41	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
28	out[8]	Output	PIN_42	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
29	out[7]	Output	PIN_43	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
30	out[6]	Output	PIN_44	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
31	out[5]	Output	PIN_45	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
32	out[4]	Output	PIN_47	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
33	out[3]	Output	PIN_48	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
34	out[2]	Output	PIN_51	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
35	out[1]	Output	PIN_52	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
36	out[0]	Output	PIN_53	4	B4_N1	3.3-V LVTTL (default)		out[19..0]	24mA (default)
37	Seg_in[15]	Input	PIN_67	4	B4_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
38	Seg_in[14]	Input	PIN_69	4	B4_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
39	Seg_in[13]	Input	PIN_70	4	B4_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
40	Seg_in[12]	Input	PIN_71	4	B4_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
41	Seg_in[11]	Input	PIN_72	4	B4_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)

42	Seg_in[10]	Input	PIN_73	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
43	Seg_in[9]	Input	PIN_74	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
44	Seg_in[8]	Input	PIN_79	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
45	Seg_in[7]	Input	PIN_80	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
46	Seg_in[6]	Input	PIN_81	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
47	Seg_in[5]	Input	PIN_86	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
48	Seg_in[4]	Input	PIN_87	3	B3_N1	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
49	Seg_in[3]	Input	PIN_92	3	B3_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
50	Seg_in[2]	Input	PIN_93	3	B3_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
51	Seg_in[1]	Input	PIN_94	3	B3_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)
52	Seg_in[0]	Input	PIN_96	3	B3_N0	3.3-V LVTTL (default)		Seg_in[15..0]	24mA (default)

编译下载到 FPGA 电路板上，然后连接实验箱的电路。

编译后如下：



3. 电路连接示意图

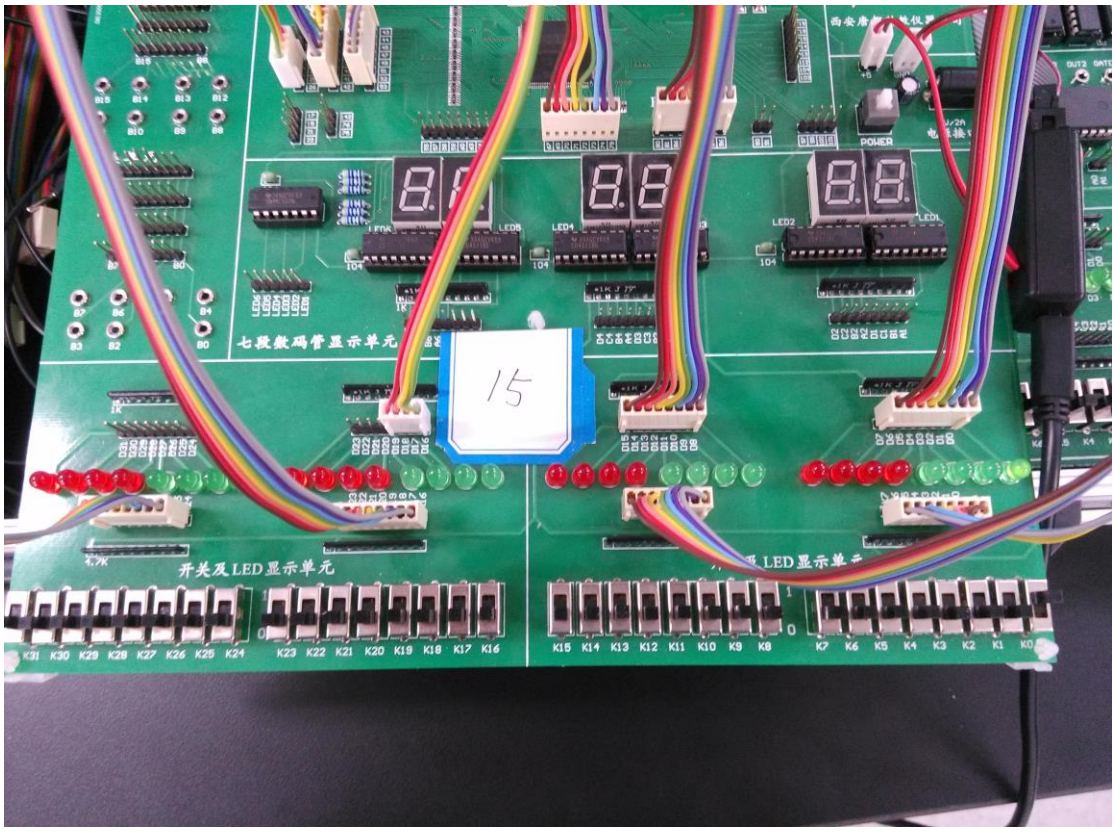
连接 FPGA 电路板上电路：

- 将 FPGA 板上的 PIN_67~PIN_96(Seg_in[15..0])接到控制开关 K31~K16 上；
- 将 FPGA 板上的 PIN_144~PIN_120(Off_in[15..0])接到 FPGA 电路板的控制开关 K15~K0 上；
- 将 FPGA 板上的 PIN_9~PIN_153(out[19..0])接到 FPGA 电路板的 LED 灯泡 L19~L0 上；

对应端口连接表			
说明	引脚	说明	引脚
FPGA 板上的段地址 Seg_in[15..0]	PIN_67~PIN_96	FPGA 电路板的控制开关	K31~K16
FPGA 板的偏移地址 Off_in[15..0]	PIN_144~PIN_120	FPGA 电路板控制开关	K15~K0
FPGA 板的输出端(物理地址 OUT[19..0]	PIN_9~PIN_153	FPGA 电路板 LED 灯泡	L19~L0

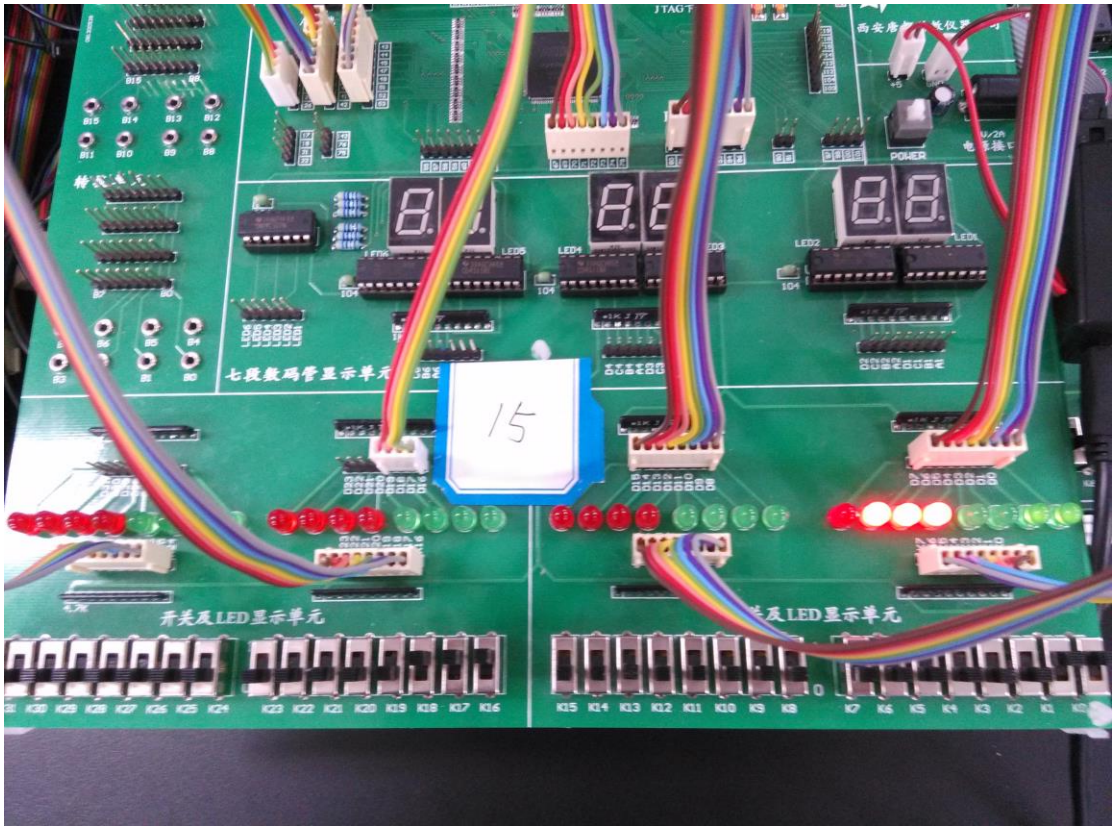
4. 实验结果

实验时拨动逻辑开关输入段地址是“0000000000000000”，拨动逻辑开关输入偏移地址是“0000000000000001”，实验结果如下：



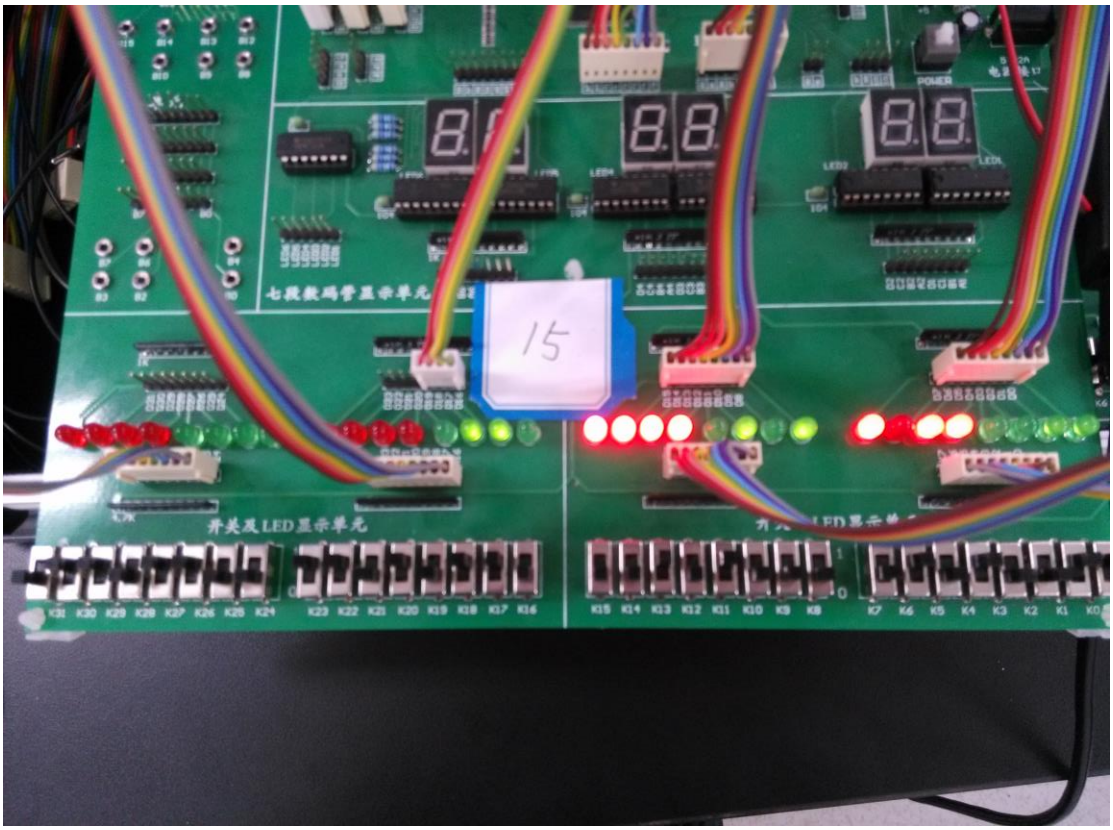
可以看出，输出物理地址是“0000000000000000001”，与预期结果一致。

实验时拨动逻辑开关输入段地址是“0000000000000111”，拨动逻辑开关输入偏移地址是“0000000000000011”，实验结果如下：



可以看出，输出物理地址是“00000111000000000011”，与预期结果一致。

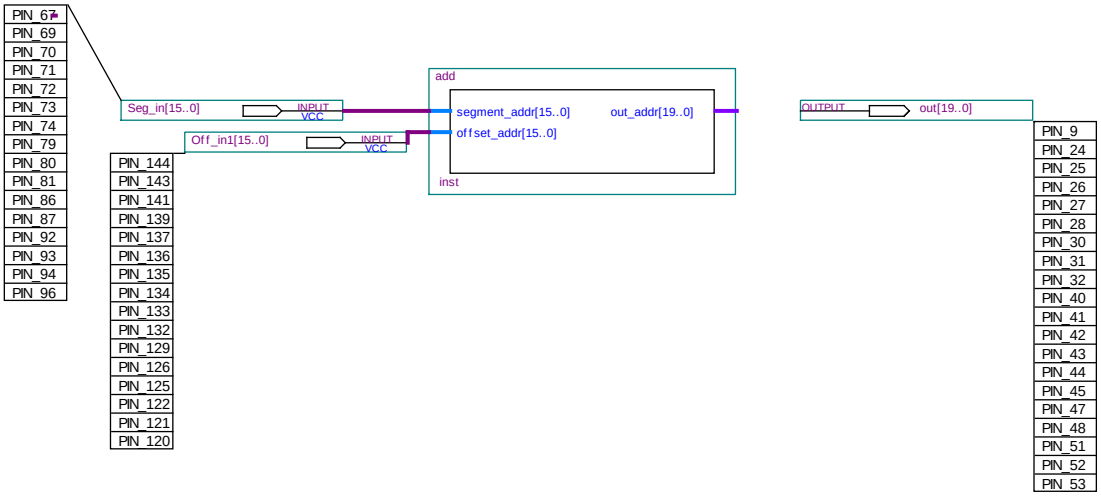
实验时拨动逻辑开关输入段地址是“0110110011010110”，拨动逻辑开关输入偏移地址是“0010100001010011”，实验结果如下：



可以看出，输出物理地址是“0110111010110110011”，与预期结果（0110110011010110*16+0010100001010011=0110111010110110011）一致，验证了程序逻辑的正确性。

四、 实验原理图和 vhdl 程序

1. 整体原理图（“transtophysics.bdf”）



2. 元部件 1（“add.vhd”）

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;

ENTITY add IS
  PORT(
       segment_addr: IN STD_LOGIC_VECTOR(15 downto 0);
       offset_addr:    IN STD_LOGIC_VECTOR(15 downto 0);
       out_addr:    OUT STD_LOGIC_VECTOR(19 downto 0)
  );
END add;

ARCHITECTURE add_architecture OF add IS
    signal a:std_logic_vector(19 downto 0);
    signal b:std_logic_vector(19 downto 0);

begin
    behavior:process(segment_addr,offset_addr) is

    variable carry_in:std_logic;
    variable carry_out:std_logic;
    variable op2:std_logic_vector(19 downto 0);

    begin
       a<= segment_addr&"0000";
```

```
    b<= "0000"&offset_addr;
    op2:=b;
    for index in 0 to 19 loop
    carry_in:=carry_out;
    out_addr(index)<=a(index) xor op2(index) xor carry_in;
    carry_out:=(a(index)and op2(index)) or (carry_in and (a(index) xor op2(index)));
  end loop;

  end process;

end add_architecture;
```

五、 实验总结

1.实验遇到的问题

这次的8086逻辑地址转换成物理地址设计实验主要在于设计一个加法电路，主要遇到的问题及解决方法如下：

- SINGAL变量不可以直接相加、异或等操作，需要转化成variable类型。
- 数据类型定义需要放在BEGIN之前，不可在程序内部定义。

2.实验感悟

这次的 8086 逻辑地址转换成物理地址设计实验相比于之前的设计实验来说逻辑稍微复杂，主要在于设计一个加法电路，我使用了循环的方法实现了这一逻辑，感觉理论知识在实际中得到了应用，同时也提高了我对于硬件逻辑电路设计的兴趣。

最后感谢老师们不厌其烦的为我解答疑惑并帮助我调试bug，我才能顺利的完成了这次实验！