实验一 译码器的实现

班级 计科 1907 姓名 杨杰 学号 201908010705

一、实验目的

熟悉 QuartusII 仿真软件的基本操作，并用 VHDL/Verilog 语言设计一个异或门，一个 3-8 译码器，一个指令译码器。

二、实验内容

1、熟悉 QuartusII 软件的基本操作，了解各种设计输入方法（原理图设计、文本设计、波形设计）

2、用 VHDL 语言设计一个异或门，最后仿真验证。

3、用 VHDL 语言设计一个 3-8 译码器，最后仿真验证。

4、用 VHDL 语言设计一个指令译码器，最后仿真验证。

三、实验方法

1、实验方法

采用基于 FPGA 进行数字逻辑电路设计的方法。

采用的软件工具是 Quartus II。

2、实验步骤

1) 异或门

1、新建，编写源代码。

(1)选择保存项和芯片类型：【File】-【new project wizard】-【next】（设置文件路径+设置 project name 为 xor2）-【next】（设置文件名 xor2.vhd—在【add】）-【properties】（type=AHDL）-【next】（family=FLEX10K；name=EPF10K10TI144-4）-【next】-【finish】

(2)新建：【file】-【new】（第二个 AHDL File）-【OK】

2、写好源代码，保存文件（xor2.vhd）。

3、编译与调试。确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功。

4、波形仿真及验证。新建一个 vector waveform file。按照程序所述插入 a,b,c 三

个节点（a、b 为输入节点，c 为输出节点）。(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all;【list】）-【>>】-【ok】-【ok】)。任意设置 a,b 的输入波形…点击保存按钮保存。（操作为：点击 name（如：A））-右击-【value】-【clock】(如设置 period=200; offset=0)，同理设置 name B（如 120，,60），保存）。然后【start simulation】，出 name C 的输出图。

5、时序仿真或功能仿真。

6、查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

2) 3-8 译码器

1、新建，编写源代码。

(1)选择保存项和芯片类型：【File】-【new project wizard】-【next】（设置文件路径+设置 project name 为 xor2）-【next】（设置文件名 xor2.vhd—在【add】）-【properties】(type=AHDL)-【next】(family=FLEX10K; name=EPF10K10TI144-4）-【next】-【finish】

(2)新建：【file】-【new】（第二个 AHDL File）-【OK】

2、写好源代码，保存文件（xor2.vhd）。

3、编译与调试。确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功。

4、波形仿真及验证。新建一个 vector waveform file。按照程序所述插入 a,b,c 三个节点（a、b 为输入节点，c 为输出节点）。(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all;【list】）-【>>】-【ok】-【ok】)。任意设置 a,b 的输入波形…点击保存按钮保存。（操作为：点击 name（如：A））-右击-【value】-【clock】(如设置 period=200; offset=0)，同理设置 name B（如 120，,60），保存）。然后【start simulation】，出 name C 的输出图。

5、时序仿真或功能仿真。

6、查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

3) 指令译码器

1、新建，编写源代码。

(1)选择保存项和芯片类型：【File】-【new project wizard】-【next】（设置文件

路径+设置 project name 为 xor2）-【next】（设置文件名 xor2.vhd—在【add】）-【properties】（type=AHDL）-【next】（family=FLEX10K；name=EPF10K10TI144-4）-【next】-【finish】

(2)新建：【file】-【new】（第二个 AHDL File）-【OK】

2、写好源代码，保存文件（xor2.vhd）。

3、编译与调试。确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功。

4、波形仿真及验证。新建一个 vector waveform file。按照程序所述插入 a,b,c 三个节点（a、b 为输入节点，c 为输出节点）。(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all;【list】）-【>>】-【ok】-【ok】)。任意设置 a,b 的输入波形…点击保存按钮保存。（操作为：点击 name（如：A））-右击-【value】-【clock】（如设置 period=200；offset=0），同理设置 name B（如 120，,60），保存）。然后【start simulation】，出 name C 的输出图。

5、时序仿真或功能仿真。

6、查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

四、实验过程

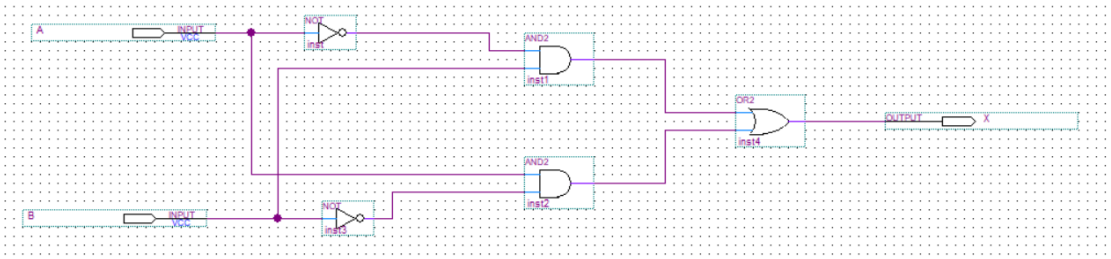
（1）异或门

1、编译过程

a）源代码如图（VHDL 设计）

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   ENTITY XOR_2 IS
4       PORT(A,B:IN STD_LOGIC;
5       F:OUT STD_LOGIC);
6   END XOR_2;
7   ARCHITECTURE ONE OF XOR_2 IS
8   SIGNAL M:STD_LOGIC_VECTOR(1 DOWNTO 0);
9   BEGIN
10  M<=A&B;
11  PROCESS(M)
12  BEGIN
13  IF M="00" THEN F<='0';
14  ELSIF M="01" THEN F<='1';
15  ELSIF M="10" THEN F<='1';
16  ELSE F<='0';
17  END IF;
18  END PROCESS;
19  END ONE;
```

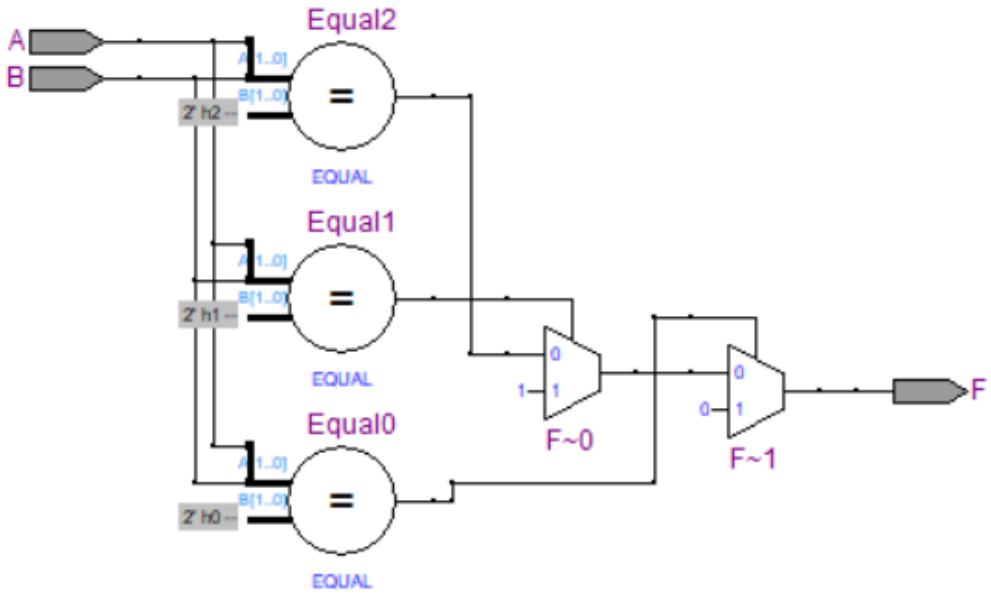
原理图如图



b)编译、调试过程

编译通过，无警告信息。

c) RTL 视图



d)结果分析及结论

输入 00 时输出 0

输入 01 时输出 1

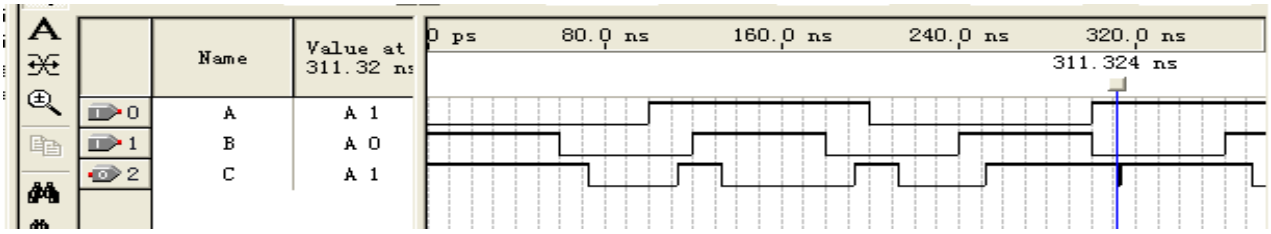
输入 10 时输出 1

输入 11 时输出 0

2、波形仿真

a)波形仿真过程（详见实验步骤）

b)波形仿真波形图



c)结果分析及结论

0-60ns：异或门，0$1=1 正确

60-70ns：A$B=0$0=0;由于有时间延迟，C 显示的是 0$1 的情况

70-100ns：A$B=0$0=0;正确

100-110ns：由于时间延迟，显示的是 0$0=0

311.324ns 分析：由于 AB 在 310ns 时同时变，造成在滞后时，出现此种情况，老

师说要避免这种情况。

3、时序仿真

a)时序仿真过程

做好上述步骤后，编译【classic timing analysis】-在 compilation report 中选择【timing analysis】-【tpd】（引脚到引脚的延时）

b)时序仿真图

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	12.900 ns	A	C
2	N/A	None	12.400 ns	B	C

c)结果分析及结论

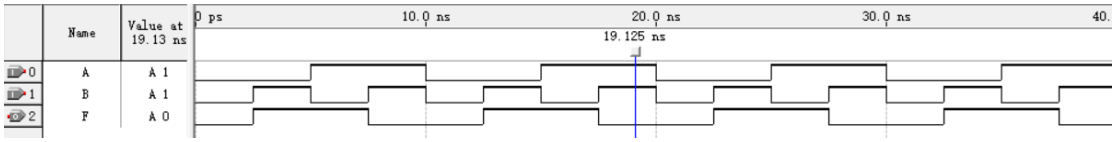
A 引脚到 C 引脚的实际 p2p 时间为 12.9ns，二 B 引脚到 C 引脚的实际 p2p 时间为 12.4ns。A 比 B 慢 0.5ns，可由于结果是由时间长的那个决定，故整体为 12.9ns。tpd (引脚到引脚的延时)

4.功能仿真

a)功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图



c)结果分析及结论

功能仿真图没有延迟，结果显示正确。

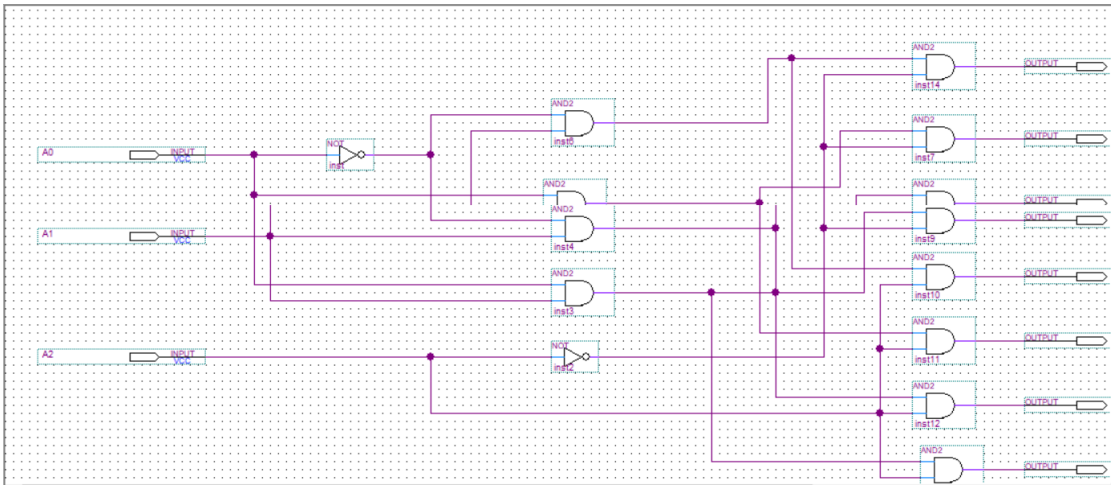
（2）3-8 译码器

1.编译过程

a）源代码如图（VHDL 设计）

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  ENTITY decoder3_8 IS
4      PORT(A0,A1,A2:IN STD_LOGIC;
5      D0,D1,D2,D3,D4,D5,D6,D7:OUT STD_LOGIC);
6  END decoder3_8;
7  ARCHITECTURE ONE OF decoder3_8 IS
8  SIGNAL X0,X1,X2:STD_LOGIC;
9  BEGIN
10 X0<=NOT A0;
11 X1<=NOT A1;
12 X2<=NOT A2;
13 D0<=X2 AND X1 AND X0;
14 D1<=X2 AND X1 AND A0;
15 D2<=X2 AND A1 AND X0;
16 D3<=X2 AND A1 AND A0;
17 D4<=A2 AND X1 AND X0;
18 D5<=A2 AND X1 AND A0;
19 D6<=A2 AND A1 AND X0;
20 D7<=A2 AND A1 AND A0;
21 END ONE;
```

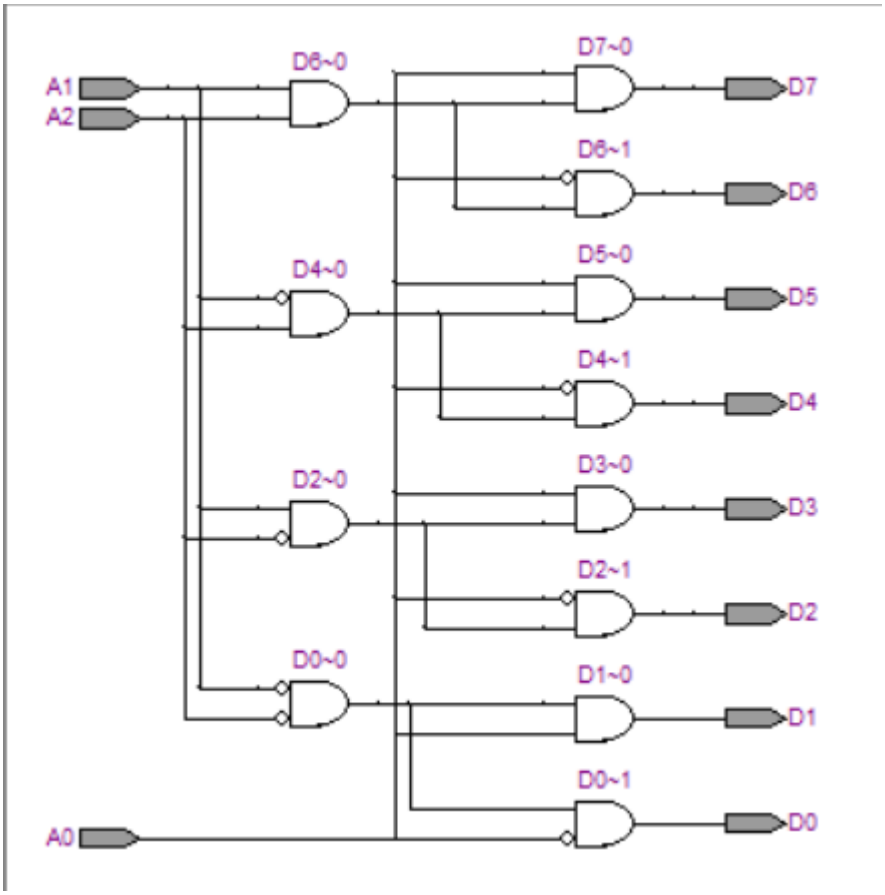
原理图如图



b)编译、调试过程

编译通过，无警告信息。

c) RTL 视图



d)结果分析及结论

当使能为 1 时，

输入 000 时输出 10000000

输入 100 时输出 00001000

输入 001 时输出 01000000

输入 101 时输出 00000100

输入 010 时输出 00100000

输入 110 时输出 00000010

输入 011 时输出 00010000

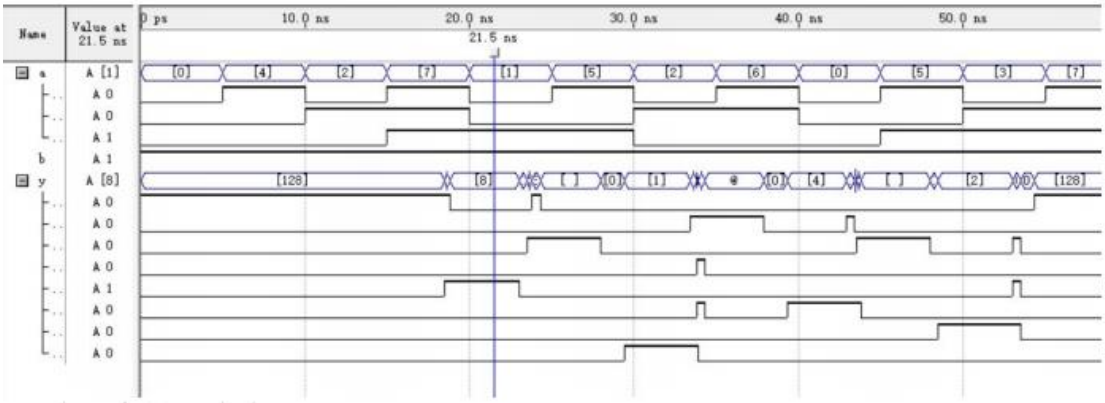
输入 111 时输出 00000001

当使能为 0 时，恒输出 00000000

2.时序仿真

a) 时序仿真过程 做好上述步骤后，编译【classic timing analysis】-在 compilation report 中选择 【timing analysis】-【tpd】（引脚到引脚的延时）

b)时序仿真波形图



c)结果分析及结论

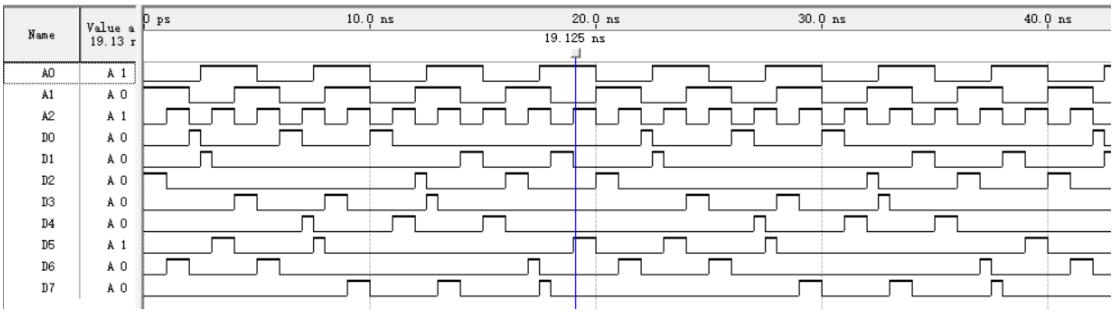
使能端 b 为 1，输出结果为实验过程中结果分析，正确

3.功能仿真

a) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图



c)结果分析及结论

功能仿真图没有延迟，结果显示正确。

（3）指令译码器

1.编译过程

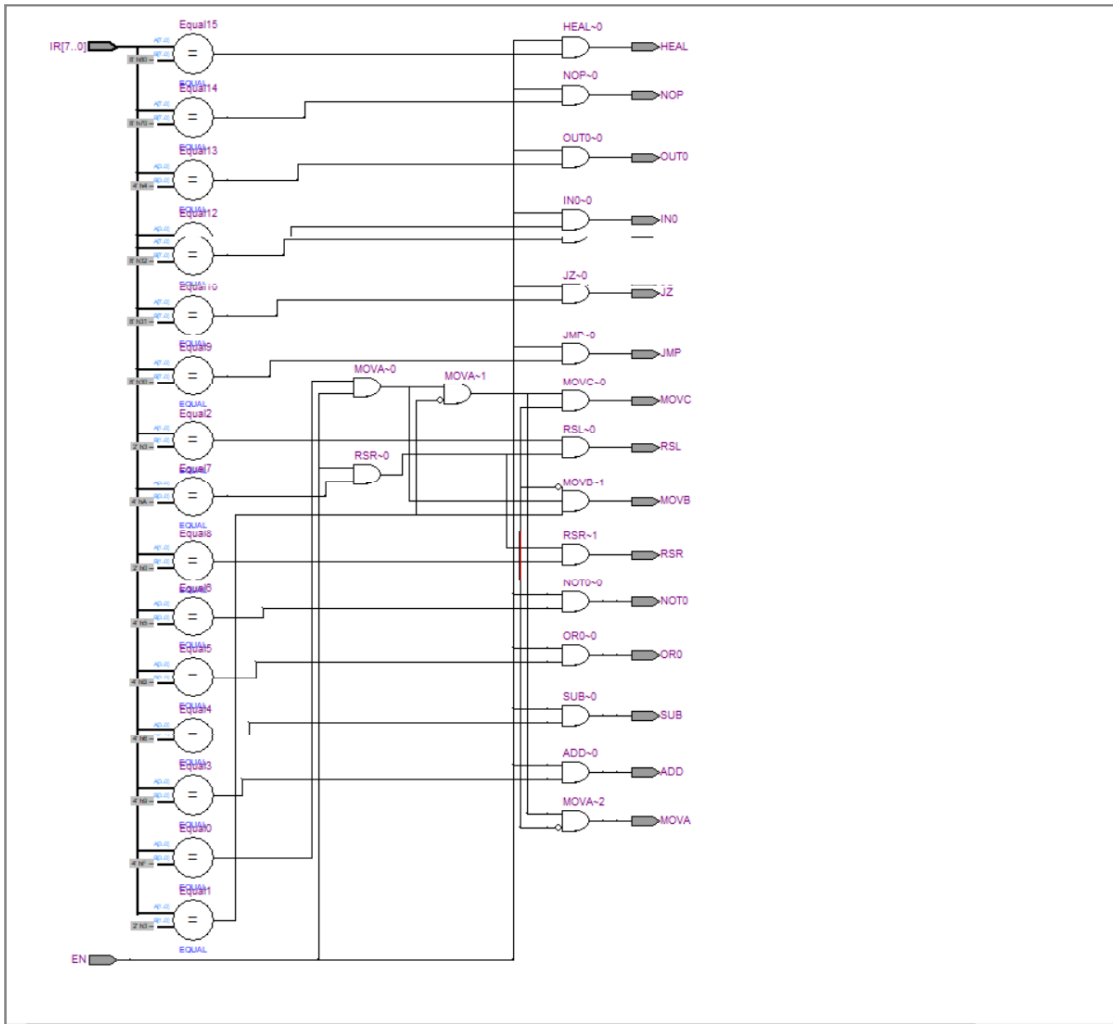
a）源代码如图（VHDL 设计）

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY decoder1 IS
    PORT(EN : IN STD_LOGIC;
         IR : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
         MOVA: OUT STD_LOGIC;
         MOVB: OUT STD_LOGIC;
         MOVC: OUT STD_LOGIC;
         ADD: OUT STD_LOGIC;
         SUB: OUT STD_LOGIC;
         OR0: OUT STD_LOGIC;
         NOT0: OUT STD_LOGIC;
         RSR: OUT STD_LOGIC;
         RSL: OUT STD_LOGIC;
         JMP: OUT STD_LOGIC;
         JZ: OUT STD_LOGIC;
         JC: OUT STD_LOGIC;
         IN0: OUT STD_LOGIC;
         OUT0: OUT STD_LOGIC;
         NOP: OUT STD_LOGIC;
         HEAL: OUT STD_LOGIC);
END decoder1;
ARCHITECTURE one OF decoder1 IS
SIGNAL code : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL R1, R2:STD_LOGIC_VECTOR(1 DOWNTO 0);
--SIGNAL MOVA_1, MOVB_1,MOVC_1, ADD_1, SUB_1, OR0_1, NOT0_1, RSR_1, RSL_1, ONE_1, JZ_1, JC_1, IN0_1, OUT0_1, NOP_1, HEAL_1:STD_LOGIC
BEGIN
    code <= IR(7 DOWNTO 4);
    R1 <= IR(3 DOWNTO 2);
    R2 <= IR(1 DOWNTO 0);
    MOVA    <= '1' WHEN(EN = '1' and code = "1111" and R1 /= "11" and R2 /= "11") ELSE '0';
    MOVB    <= '1' WHEN(EN = '1' and code = "1111" and R1 = "11" and R2 /= "11") ELSE '0';
    MOVC    <= '1' WHEN(EN = '1' and code = "1111" and R1 /= "11" and R2 = "11") ELSE '0';
    ADD     <= '1' WHEN(EN = '1' and code = "1001") ELSE '0';
    SUB     <= '1' WHEN(EN = '1' and code = "0110") ELSE '0';
    OR0     <= '1' WHEN(EN = '1' and code = "1011") ELSE '0';
    NOT0    <= '1' WHEN(EN = '1' and code = "0101") ELSE '0';
    RSR     <= '1' WHEN(EN = '1' and code = "1010" and R2 = "00") ELSE '0';
    RSL     <= '1' WHEN(EN = '1' and code = "1010" and R2 = "11") ELSE '0';
    JMP     <= '1' WHEN(EN = '1' and IR = "00110000") ELSE '0';
    JZ      <= '1' WHEN(EN = '1' and IR = "00110001") ELSE '0';
    JC      <= '1' WHEN(EN = '1' and IR = "00110010") ELSE '0';
    IN0     <= '1' WHEN(EN = '1' and code = "0010") ELSE '0';
    OUT0    <= '1' WHEN(EN = '1' and code = "0100") ELSE '0';
    NOP     <= '1' WHEN(EN = '1' and IR = "01110000") ELSE '0';
    HEAL    <= '1' WHEN(EN = '1' and IR = "10000000") ELSE '0';
END one;
```

b)编译、调试过程

编译通过，无警告信息。

c) RTL 视图



d)结果分析及结论

输入 1111 R1 R2 执行 MOV M 操作，此时 mova=1

输入 1111 11 R2 执行 MOV R1，M 的操作此时 movb=1

输入 1111 R1 11 执行 MOVC 的操作，此时 movc=1

输入 1001 R1 R2 执行（R1）+（R2）→ R1 add=1

输入 0110 R1 R2 执行（R1）-（R2）→ R1 sub=1

输入 1011 R1 R2 执行（R1）∨（R2）→ R1 or0=1

输入 0101 R1 执行 /（R1）→ R1 not0=1

输入 1010 R1 00 执行循环右移一位 rsr=1

输入 1010 R1 00 执行循环右移一位 rsl=1

输入 0001 00 00，0001 00 01，jmp 为 1 或 jz 为 1 或 jc 为 1

输入 0010 R1XX 执行输入操作 ino=1

输入 0100 R1XX 执行输出操作 out0=1

输入 01110000 执行 NOP 操作 nop=1

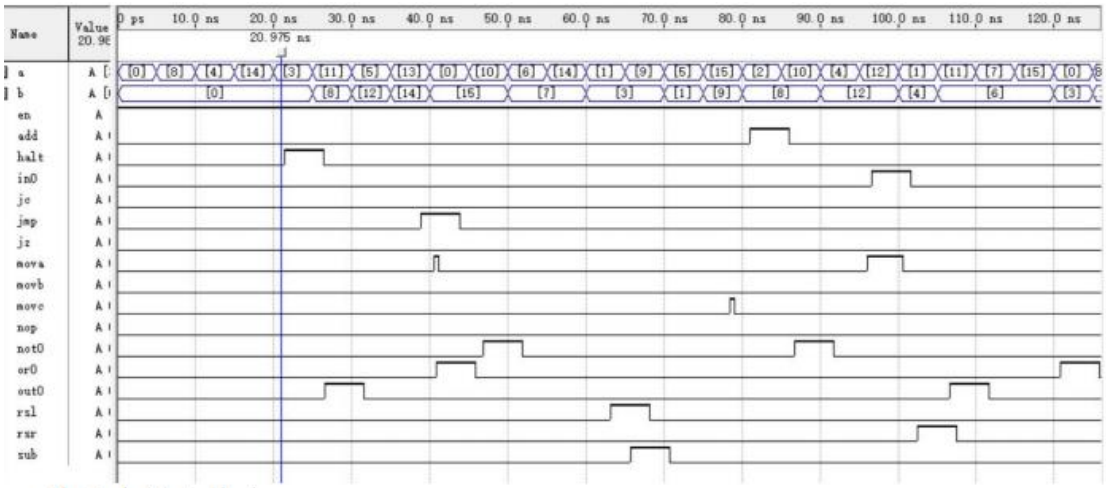
输入 10000000 执行停机操作 halt=1

2.时序仿真

a)时序仿真过程

做好上述步骤后，编译【classic timing analysis】-在 compilation report 中选择【timing analysis】-【tpd】（引脚到引脚的延时）

b)时序仿真波形图



c)结果分析及结论

使能端 en 为 1，输出结果符合实验过程中结果分析，正确

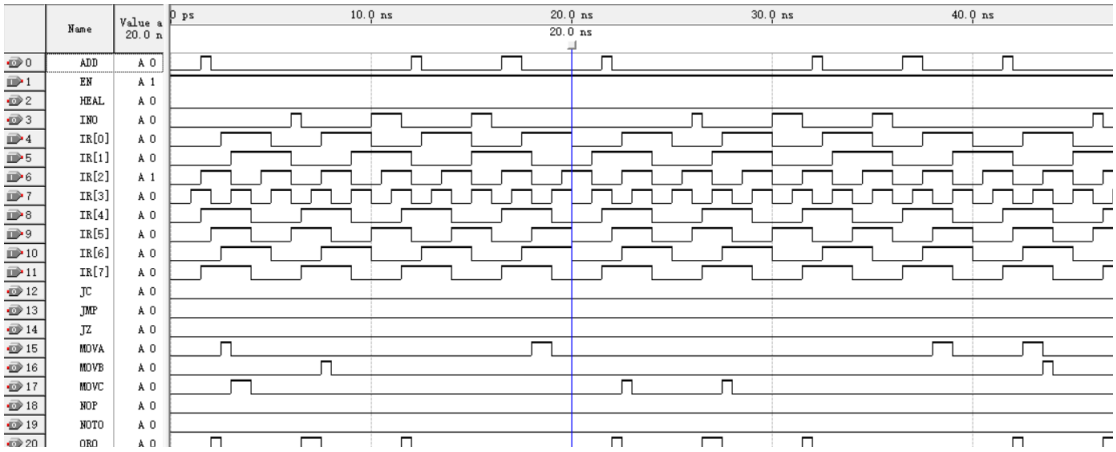
代码正确

3.功能仿真

a)功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图



c)结果分析及结论

功能仿真图没有延迟，结果显示正确

五、实验结论

1、思考题

1)在日常生活中，我们哪些场所会用到译码器？

译码器(decoder)是一类多输入多输出组合逻辑电路器件，其可以分为：变量译码和显示译码两类。变量译码器一般是一种较少输入变为较多输出的器件，常见的有 n 线-2^n 线译码和 8421BCD 码译码两类；显示译码器用来将二进制数转换成对应的七段码，一般其可分为驱动 LED 和驱动 LCD 两类。

译码器的种类很多，但它们的工作原理和分析设计方法大同小异，其中二进制译码器、二-十进制译码器和显示译码器是三种最典型，使用十分广泛的译码电路。

二进制码译码器，也称最小项译码器，N 中取一译码器，最小项译码器一般是将二进制码译为十进制码；

代码转换译码器，是从一种编码转换为另一种编码；

显示译码器，一般是将一种编码译成十进制码或特定的编码，并通过显示器件将译码器的状态显示出来。

在日常生活中，我们在 LED 数码管、舞台上的显示译码器、在工业控制中小型电子设备也会用到译码器。

2)总结 VHDL 描述译码器电路的方法和常用语句?

第一种按照真值表八种情况对应八种输出在语句中的表达就是当出现 xxx 的时候就输出 xxxxxxxx 的结果;第二种按逻辑电路来写代码将 a 非 b 非信号都表示出来八种组合对应相应的输出，定义 8 个信号分别对应哪八种组合。

3)比较原理图方式和 VHDL 方式设计组合逻辑电路的方法、步骤和优缺点。

原理图的输入和 VHDL 文本在原理上面是一样的，等价的。

原理图输入更加直观，能够更清楚地表示该电路输入信号与输出信号之间的逻辑关系，省事又直观得设计相应的组合逻辑电路，拓扑阅读性好，对于小规模的设计来讲，更加快捷简便;

VHDL 语言，其功能强大，灵活性强，更加容易修改和阅读，逻辑表达比较清晰，模块化也不错，对于较大规模的设计来讲，移植更加方便。但电路采用高级的简明结构 VHDL 描述，意味着放弃了对电路门级实现定义的控制，由综合工具产生的逻辑实现效果有时不优化，采用工具的不同导致综合质量不一样。VHDL 语言设计组合逻辑电路先进行系统设计，再进行逻辑设计，功能仿真，逻辑综合，布局布线，时序仿真。

2、 实验总结与实验心得

这是第一次利用 VHDL 语言编写程序，VHDL 语言是我接触到的第一门硬件描述语言。开始的时候并不是很理解 VHDL 语言，导致编写程序很吃力。在查阅有关资料后，慢慢的开始理解了 VDHL 语言。本实验要求通过 VHDL 语言设计一个异或门，一个 3-8 译码器，一个指令译码器。在实验之前一定要对这三个元器件的原理有一个基础的了解，这样才能顺利完成实验。VHDL 语言有三种不同的描述风格，分别是：结构描述、数据流描述、行为描述。在本实验中我采用行为描述。

实验之前一定要先有一个预先的设计，比如可以用什么方式来实现该功能，每种方式有何种优缺点，为了结果的准确性与严谨性本实验采用何种方式更好。比如此次实验的指令译码器，最初直接使用了枚举法，没有考虑到枚举法会出现 mova、movb、movc 同时为 1 的情况，而这种情况是错误的。助教指出了这个

错误，最后使用了 if-else 的方式重新实现。在完成代码的过程中，一要仔细，减少基础性的失误；二要使得代码尽可能的简洁，增加代码的可读性，提高代码的质量。