实验一 译码器的实现（实验报告格式案例）

班级 软件 1801 姓名 肖云杰 学号 201826010113

一、实验目的

1. 熟悉多路复用器以及模型机的工作原理。
2. 学会使用 VHDL 语言设计多路复用器。
3. 掌握 generic 的使用，能运用设计参数化多路复用器。
4. 学会使用 VHDL 语言设计模型机控制信号产生逻辑。

二、实验内容

1、用 VHDL 语言设计模型机的 8 重 3-1 多路复用器；

2、设计参数化多路复用器，调用该参数化多路复用器定制为 8-1 多路复用器。

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。

设计方法采用的是 VHDL 设计法。

2、实验步骤

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 mux83_1）-【next】（设置文件名 mux83_1.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。

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

A 8 重 3-1 多路复用器

1、编译过程

a）源代码如图（VHDL 设计）

mux83_1.vhd | mux83_1.vwf

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  entity mux83_1 is
4  port(d0,d1,d2:in STD_LOGIC_VECTOR(7 downto 0);
5           sel :in STD_LOGIC_VECTOR(1 downto 0);
6           dout:out STD_LOGIC_VECTOR(7 downto 0));
7  end mux83_1;
8  architecture rtl of mux83_1 is
9  begin
10     dout <=
11     d0 when sel="00" else
12     d1 when sel="01" else
13     d2;
14 end rtl;
15
```

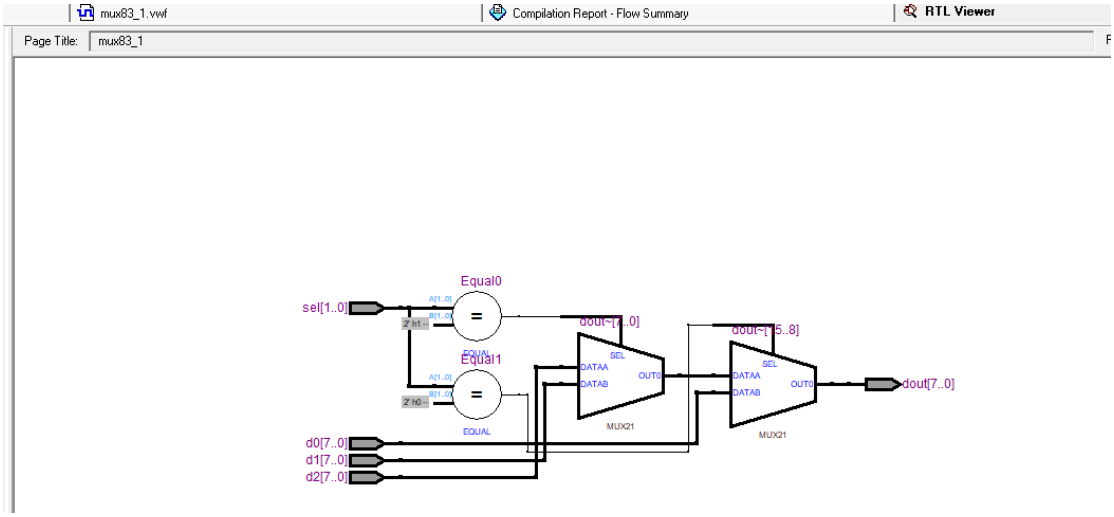
b)编译、调试过程

Warning: No exact pin location assignment(s) for 34 pins of 34 total pins

Warning: The Reserve All Unused Pins setting has not been specified, and will default to 'As output driving ground'.

（全部都是管脚未分配问题）

c) RTL 视图



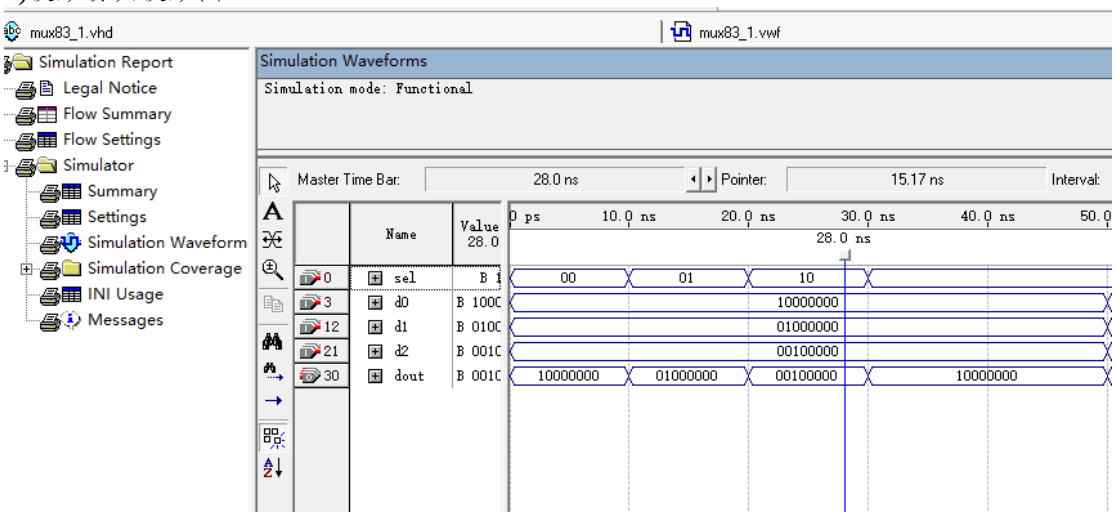
d)结果分析及结论

VHDL 文件实现的文件功能基本正确，除了管脚未分配问题之外，没有报错

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

0-10ns：选择信号为 00，选择 d0 输出，正确

10-20ns：选择信号为 01，选择 d1 输出，正确

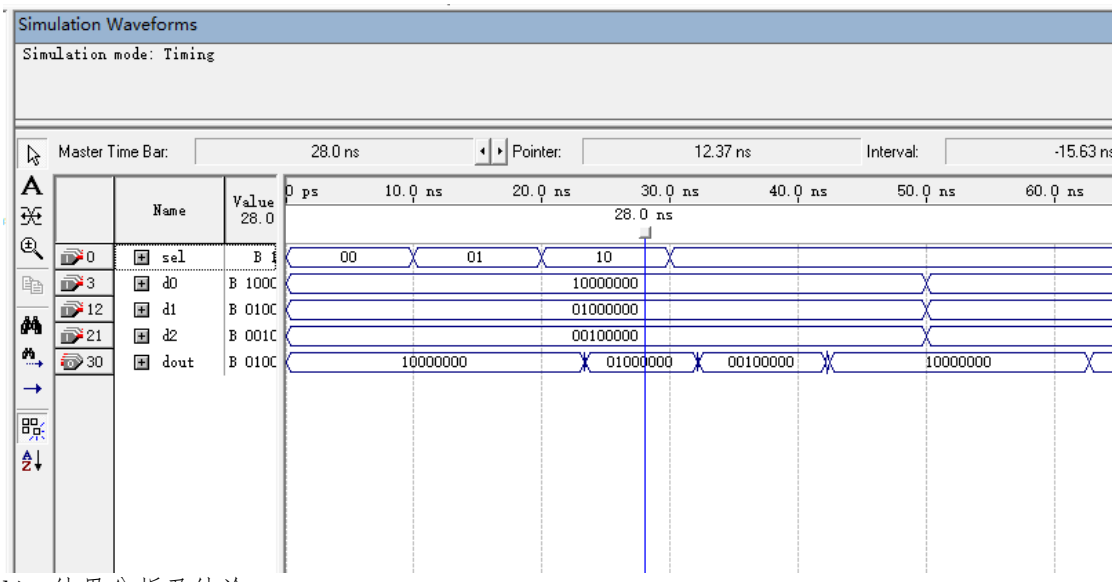
20-30ns：选择信号为 10，选择 d2 输出，正确

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



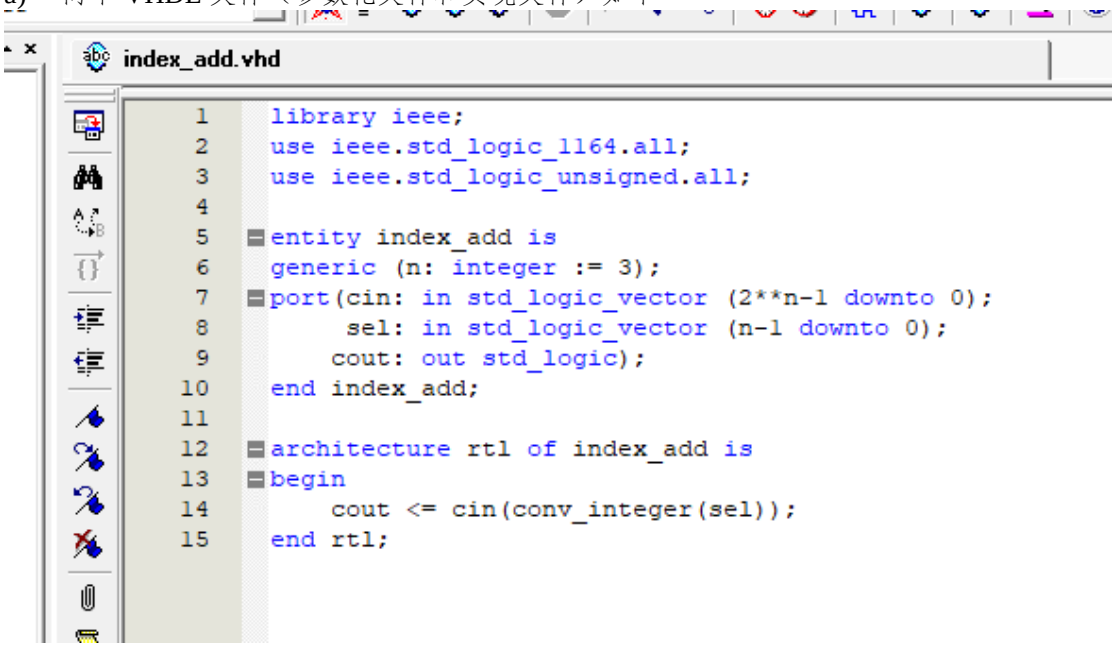
b) 结果分析及结论

引脚之间，存在信号传输的延迟，信号传输的延迟大约是 14ns

B 参数化 8-1 多路复用器

1、 **编译过程**

a) 两个 VHDL 文件（参数化文件和实现文件）如下



dex_add.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  entity dex_add is
5  port(d:in std_logic_vector(7 downto 0);
6    se:in std_logic_vector(2 downto 0);
7    dout:out std_logic);
8  end dex_add;
9  architecture exa of dex_add is
10     component index_add is
11         generic(n:integer:=3);
12             port(cin: in std_logic_vector (2**n-1 downto 0);
13                  sel: in std_logic_vector (n-1 downto 0);
14                 cout: out std_logic);
15     end component;
16 begin
17     g0:index_add generic map(n=>3) port map( cin=>d , sel=>se , cout=>dout);
18 end exa;
```

b) 编译过程

Warning: Using design file index_add.vhd, which is not specified as a design file for the current project, but contains definitions for 2 design units and 1 entities in project

Info: Found design unit 1: index_add-rtl

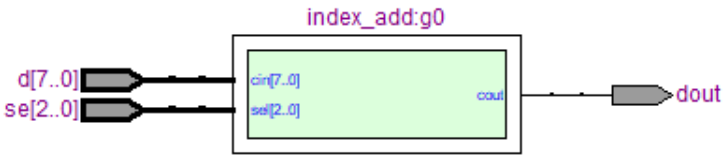
Info: Found entity 1: index_add

Warning: No exact pin location assignment(s) for 12 pins of 12 total pins

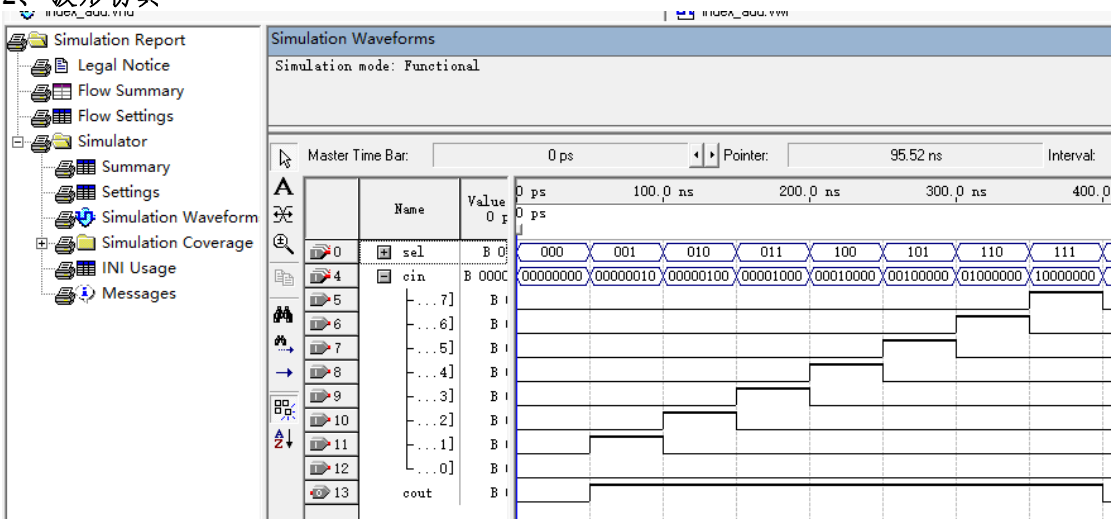
Warning: The Reserve All Unused Pins setting has not been specified, and will default to 'As output driving ground'.

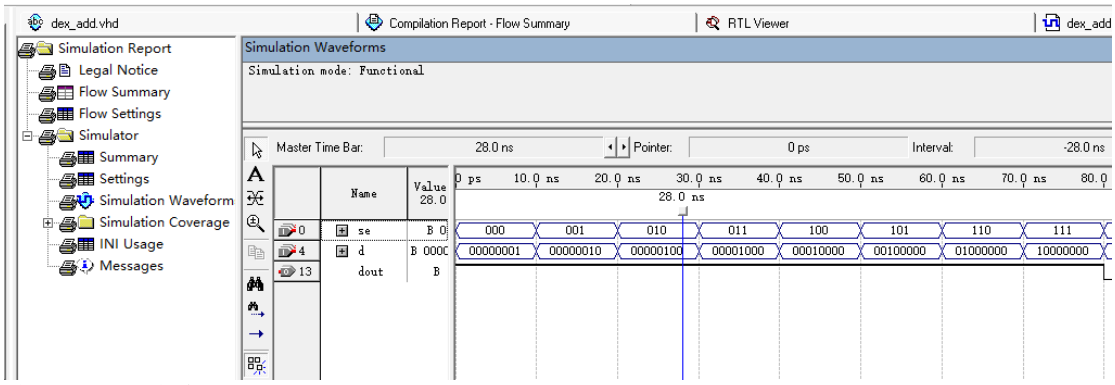
（共三个编译警告，分别是文件引用问题和管脚分配问题）

c) RTL 视图



2、波形仿真



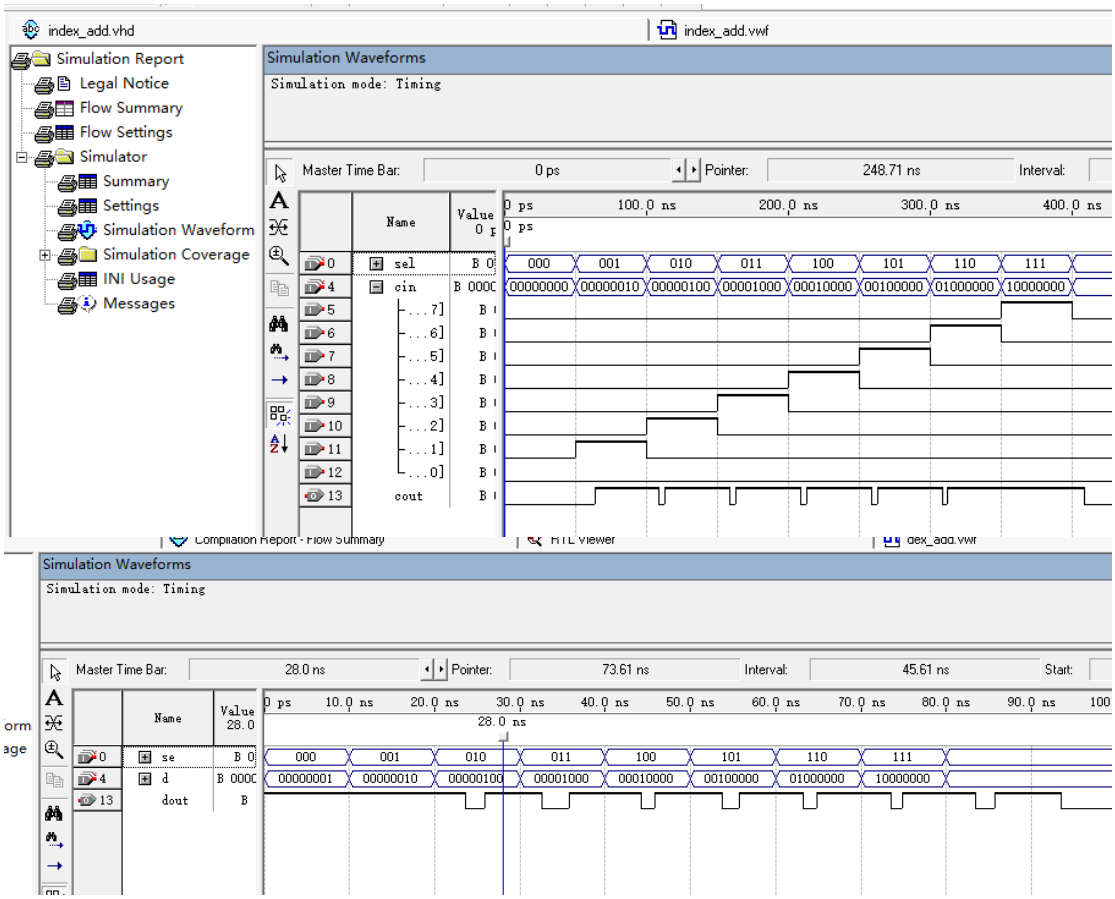


结果分析及结论

第一张图参数化多路复用器，根据 sel 信号，cout 正确选择了 cin 中相应位数的值，检验正确

第二张图定制化 8-1 多路复用器，根据 se 信号，dout 正确选择的 d 中相应的值，检验正确

3、时序仿真



结果分析及结论

在两图的时序仿真中，波形都没有太大的变化，两种方式都出现了约 14ns 的传播延迟。分析延迟，可能是管脚之间信号传输的延迟。

五、实验结论

1、思考题

a) 多路复用器的实现方法很多，请总结两种以上实现方法。

1. 第一种方法为常见的针对问题编程解决 8 重 3-1 多路选择器，根据需要的位数，输入正确的选择信号，选择相应的数据进行输出
2. 第二种方法为参数化定制方法，只需要写出相应的参数化电路然后在需要的时候引入相应文件，再进行端口的映射即可

第一种方法比较容易实现，第二种方法在用到多个多路复用器时使用比较多，不过前期编程可能会比较麻烦。

b) 总结 VHDL 语言描述多路复用器的方法和常用语句。

when else

If then; elsif then; end if;

c) 如何产生正确的控制信号以及具体的编程实现

一般采用时钟信号的来控制控制信号，每次时钟信号沿上升沿变化，即改变一个控制信号，根据控制信号实现正确的逻辑。(计数器)

这里采用一个模八计数器作为例子

mod8.vhd

```
1  library IEEE;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5  entity mod8 is
6  port(
7       clk:in STD_LOGIC;
8       c_out:out STD_LOGIC_VECTOR(2 downto 0));
9  end mod8;
10 architecture mo8 of mod8 is
11 signal iq:integer;
12 begin
13      process(clk)
14      begin
15      if clk'event and clk='1' then
16       if iq=8 then iq<=0;
17       else iq<=iq+1;
18       end if;
19      end if;
20      c_out <= conv_std_logic_vector(iq,3);
21      end process;
22 end mo8;
```

2、 实验总结与实验心得

实验过程中，在第一个文件八重 3-1 多路选择器中，因为写过多个相似的程序，没有感觉到什么困难。

但是在做第二个参数化定制多路选择器中，因为是第一次接触这种问题，感觉到了明显的难度和陌生，又因为参数化定制中，参数的输入问题，进行了两次问题的重写。并且在最后实例化的过程中遇到了障碍，最后是询问同学，打开了自己的思路才得以决绝该问题。

通过这次实验，我了解到，同一个器件（多路选择器），也会有多种实现方法，要灵活掌握集中实现方法，并且选择合适的方法进行解题。