实验二 多路复用器与加法器的实现

班级 计科 1903 姓名 陈旭 学号 201914020128

一、实验目的

1. 熟悉多路复用器、加法器的工作原理。
2. 学会使用 VHDL 语言设计多路复用器、加法器。
3. 掌握 generic 的使用，设计 n-1 多路复用器。
4. 兼顾速度与成本，设计行波加法器和先行进位加法器。

二、实验内容

1. 用 VHDL 语言设计 8 重 3-1 多路复用器。
2. 用 VHDL 语言设计 n-1 多路复用器，调用该 n-1 多路复用器定制为 8-1多路复用器。
3. 用 VHDL 语言设计 4 位行波进位加法器。
4. 用 VHDL 语言设计 4 位先行进位加法器。

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。
- 采用基本逻辑门电路和组合逻辑电路实现 8 重 3-1 多路复用器、n-1 多路复用器，并调用该 n-1 多
  路复用器定制为 8-1 多路复用器、4 位行波进位加法器、4 位先行进位加法器。
- 采用的软件工具是 Quartus II。

2、实验软件操作步骤

- **8 重 3-1 多路复用器**

1、新建，编写源代码。

(1)选择保存项和芯片类型：【File】-【new project wizard】-【next】（设置文件路径为 C:\Users\86150\Desktop\mylearn\vscode-c++\logic and computer design fundamentals\experiment2\mux_8_3_1 设置 project name 为 mux_8_3_1）-【next】（设置文件名 mux_8_3_1）-【next】（设置芯片类型为【cyclone-EP1CT144C8】）-【finish】

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

2、写好源代码，保存文件（mux_8_3_1.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，编译结果有警告，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 inputs0, inputs1, inputs2, selinputs, outputs 四个节点（inputs0, inputs1, inputs2 为八位输入节点向量，selinputs 为两位输出节点向量， outputs 为八位输出节点向量）。(操作为：右击 -【insert】-【insert

node or bus】-【node finder】(pins=all;【list】)-【>>】-【ok】-【ok】)。任意设置 inputs0, inputs1, inputs2, selinputs 的输入波形…点击保存按钮保存。(操作为：点击 name（如：enable)）-右击-【value】-【count】(如设置 binary；start value=0; count value=5.0ns)，同理设置 name b（如 0, 1, 5)，保存)。然后【start simulation】，出 name C 的输出图。

5、时序仿真和功能仿真。

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

- **n-1 多路复用器定制的 8-1 多路复用器**

1. 新建，编写源代码。

(1)选择保存项和芯片类型:【File】-【new project wizard】-【next】(设置文件路径为 C:\Users\86150\Desktop\mylearn\vscodec++\logicandcomputerdesignfundamentals\experiment2\generic_mux 设置两个文件的 project name 为 generic1(类属实体声明文件)以及 generic_mux_8_1(调用类属实体的顶层实体文件))-【next】(设置文件名 generic1、generic_mux_8_1)-【next】(设置芯片类型为【cyclone-EP1CT144C8】)-【finish】

(2). 新建:【file】-【new】(VHDL File)-【OK】

2. 写好源代码，保存文件（generic1.vhd、generic_mux_8_1.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，编译结果有警告，编译成功。
4. 波形仿真及验证。新建 generic_mux_8_1 的 vector waveform file。按照程序所述插入节点 inputs, output, selects。(inputs, selects 为输入节点向量，output 为输出节点)。(操作为：右击 -【insert】-【insert node or bus】-【node finder】(pins=all;【list】)-【>>】-【ok】-【ok】)。按顺序设置 inputs 和 selects 的输入波形…点击保存按钮保存。(操作为:点击 name(如:inputs))-右击-【value】-【count】(如设置 binary;start value=00000000; end value=11111111;count value=2.0ns)，同理设置 name b（如 000)，保存)。然后【start simulation】，出 name C 的输出图。
5. 功能仿真判断代码的实现无误后时序仿真。
6. 查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】

四、实验过程

1、编译过程

a）源代码（VHDL 设计）如下

- **8 重 3-1 多路复用器**

```
library ieee;
use ieee.std_logic_1164.all;

entity mux_8_3_1 is
    port(
        inputs0: in std_logic_vector(7 downto 0);
        inputs1: in std_logic_vector(7 downto 0);
        inputs2: in std_logic_vector(7 downto 0);
        selinputs: in std_logic_vector(1 downto 0);
        outputs: out std_logic_vector(7 downto 0)
    );
end mux_8_3_1;
```

```
architecture mux_8_3_1 of mux_8_3_1 is
    begin
        outputs<=inputs0 when selinputs="00" else
        inputs1 when selinputs="01" else
        inputs2 when selinputs="10"else
        "00000000";
end mux_8_3_1;
```

- **n-1 多路复用器定制的 8-1 多路复用器**
  - **generic1.vhd(类属实体声明文件)**

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity generic1 is
    --generic (n:integer:=3);
    generic (n:integer);
    port(
        genein: in std_logic_vector(2**n-1 downto 0);
        genesel: in std_logic_vector(n-1 downto 0);
        geneout: out std_logic
    );
    end generic1;

architecture generic1 of generic1 is
    begin
        geneout<=genein(conv_integer(genesel));
    end generic1;
```

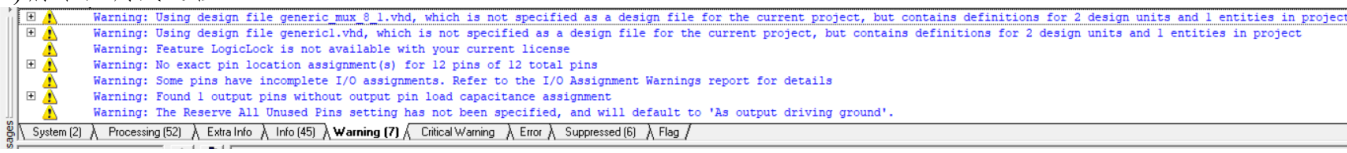
  - **generic_mux_8_1(调用类属实体的顶层实体文件)**

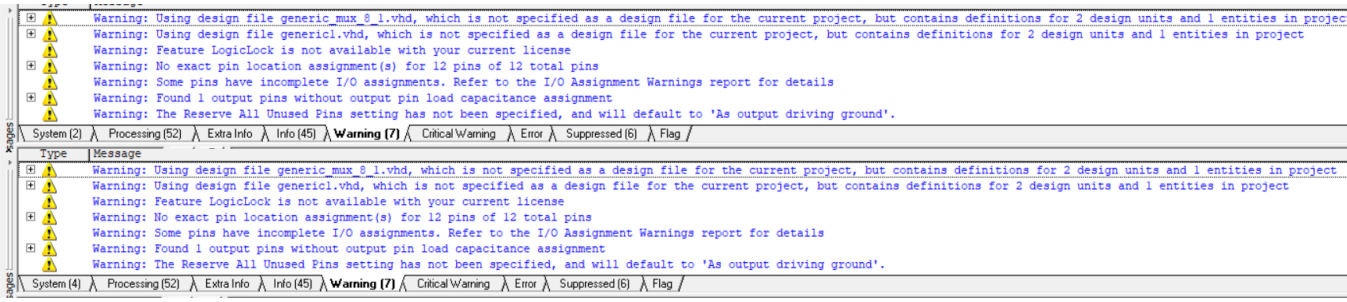
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity generic_mux_8_1 is
    port(
        inputs: in std_logic_vector(7 downto 0);
        output: out std_logic;
        selects: in std_logic_vector(2 downto 0)
    );
end generic_mux_8_1;
```

```
architecture generic_mux_8_1 of generic_mux_8_1 is
    component generic1 is
        generic (n:integer:=3);
        port(
            genein: in std_logic_vector(2**n-1 downto 0);
            genesel: in std_logic_vector(n-1 downto 0);
            geneout: out std_logic
        );
    end component;
    begin
        g1:generic1 generic map(n=>3) port map(genein=>inputs, gene
           out=>output, genesel=>selects);
    end generic_mux_8_1;
```

b)编译、调试过程

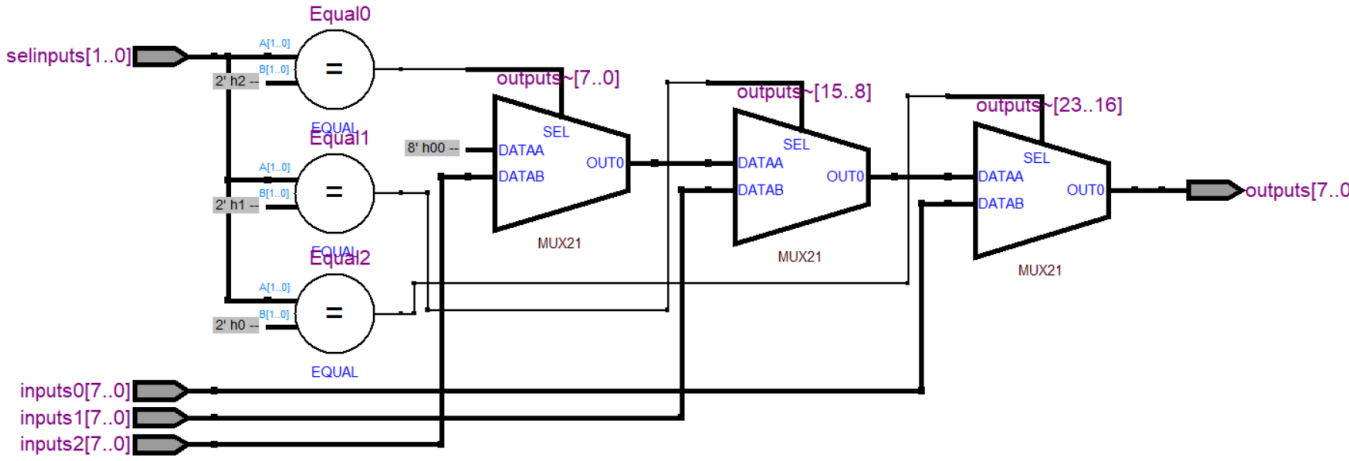




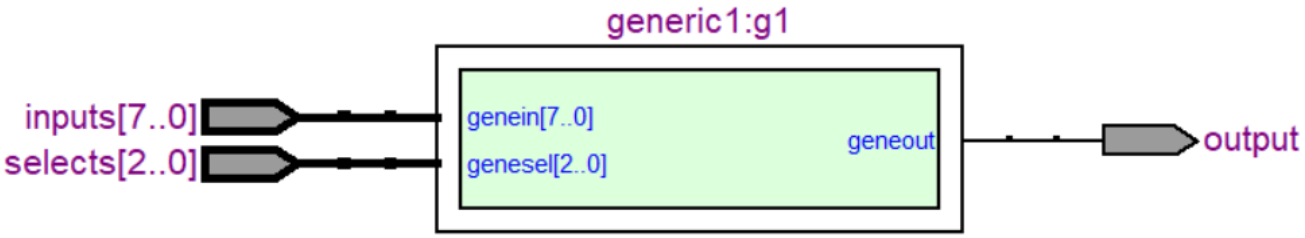
自上至下依次为 mux_8_3_1.vhd、generic1.vhd、generic_mux_8_1.vhd 的编译器提示信息。
三者编译器均给出警告，但均能够编译通过。
三者资源消耗基本都趋近于 0.

c) RTL 视图

- 8 重 3-1 多路复用器



- 调用类属实体实现的 8-1 多路复用器



d)结果分析及结论

- 8 重 3-1 多路复用器

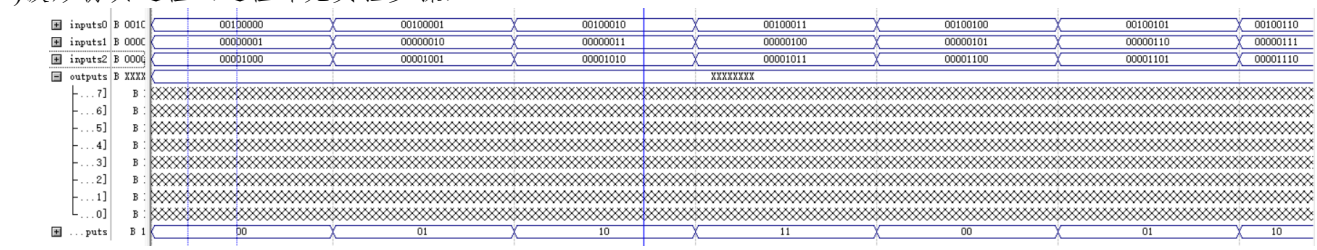
  当输入selinputs为 0 0输出inputs0对应的向量值
  当输入selinputs为 0 1输出inputs1对应的向量值
  当输入selinputs为 1 0输出inputs2对应的向量值
  当输入selinputs为 1 1输出00000000(无效选择输入)
- 调用类属实体实现的 8-1 多路复用器

  当输入 select 为 000 时，输出向量 inputs[0]对应的元素
  当输入 select 为 001 时，输出向量 inputs[1]对应的元素
  当输入 select 为 010 时，输出向量 inputs[2]对应的元素
  当输入 select 为 011 时，输出向量 inputs[3]对应的元素
  当输入 select 为 100 时，输出向量 inputs[4]对应的元素
  当输入 select 为 101 时，输出向量 inputs[5]对应的元素
  当输入 select 为 110 时，输出向量 inputs[6]对应的元素
  当输入 select 为 111 时，输出向量 inputs[7]对应的元素

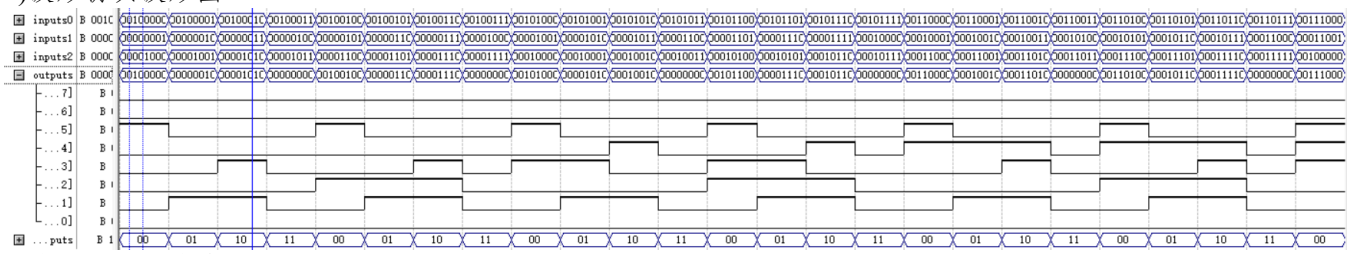
2、 波形仿真

- 8 重 3-1 多路复用器

a)波形仿真过程（过程详见实验步骤）



b)波形仿真波形图



c)结果分析及结论

以第一个周期为例

输入向量 selinputs 为 00 时，输出 inputs0 对应值，为 00100000；

输入向量 selinputs 为 01 时，输出 inputs1 对应值，为 00000010；

输入向量 selinputs 为 10 时，输出 inputs2 对应值，为 00001010；

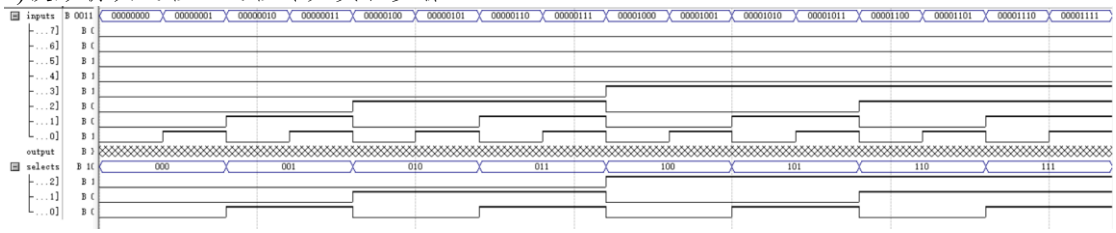
输入向量 selinputs 为 11 时，为无效的输入选择值，输出 00000000。

其他任意周期均符合上述规律。

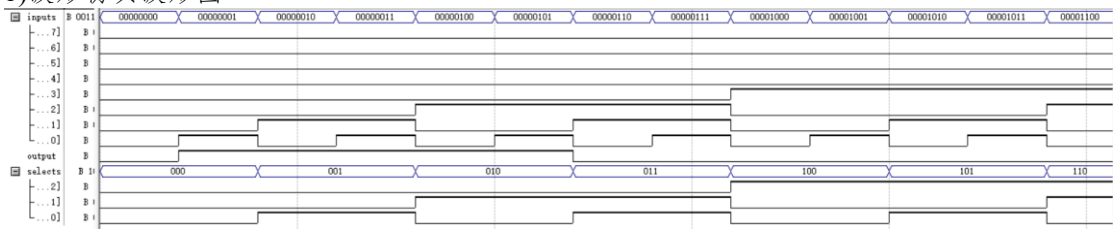
结果正确

- 调用类属实体实现的 8-1 多路复用器

a)波形仿真过程（过程详见实验步骤）



b)波形仿真波形图



c)结果分析及结论

以 select 输入从 000 变化到 110 的波形段为例。

select 输入为 000 时，输出 inputs[0]的值，在图中 output 的值先为 0，然后变化为 1

select 输入为 001 时，输出 inputs[1]的值，在图中 output 的值保持为 1

select 输入为 010 时，输出 inputs[2]的值，在图中 output 的值保持为 1

select 输入为 011 时，输出 inputs[3]的值，在图中 output 的值保持为 0

select 输入为 100 时，输出 inputs[4]的值，在图中 output 的值保持为 0

select 输入为 101 时，输出 inputs[5]的值，在图中 output 的值保持为 0

select 输入为 110 时, 输出 inputs[6]的值，在图中 output 的值保持为 0
结果正确

3、 时序仿真

- 8 重 3-1 多路复用器

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	10.407 ns	selinputs[1]	outputs[0]
2	N/A	None	10.393 ns	inputs1[0]	outputs[0]
3	N/A	None	10.133 ns	selinputs[0]	outputs[0]
4	N/A	None	9.882 ns	inputs2[0]	outputs[0]
5	N/A	None	9.729 ns	selinputs[1]	outputs[7]
6	N/A	None	9.480 ns	selinputs[1]	outputs[3]
7	N/A	None	9.454 ns	selinputs[0]	outputs[7]
8	N/A	None	9.386 ns	inputs0[0]	outputs[0]
9	N/A	None	9.205 ns	selinputs[0]	outputs[3]
10	N/A	None	9.167 ns	inputs0[7]	outputs[7]
11	N/A	None	9.118 ns	selinputs[1]	outputs[5]
12	N/A	None	9.111 ns	inputs1[5]	outputs[5]
13	N/A	None	9.080 ns	inputs1[7]	outputs[7]
14	N/A	None	8.895 ns	inputs2[7]	outputs[7]
15	N/A	None	8.846 ns	selinputs[0]	outputs[5]
16	N/A	None	8.812 ns	selinputs[1]	outputs[6]
17	N/A	None	8.810 ns	inputs1[3]	outputs[3]
18	N/A	None	8.691 ns	selinputs[1]	outputs[4]
19	N/A	None	8.676 ns	inputs0[3]	outputs[3]
20	N/A	None	8.609 ns	inputs2[3]	outputs[3]

21	N/A	None	8.537 ns	selinputs[0]	outputs[6]
22	N/A	None	8.480 ns	inputs2[6]	outputs[6]
23	N/A	None	8.473 ns	selinputs[1]	outputs[2]
24	N/A	None	8.419 ns	selinputs[0]	outputs[4]
25	N/A	None	8.297 ns	selinputs[1]	outputs[1]
26	N/A	None	8.267 ns	inputs0[5]	outputs[5]
27	N/A	None	8.198 ns	selinputs[0]	outputs[2]
28	N/A	None	8.075 ns	inputs2[5]	outputs[5]
29	N/A	None	8.066 ns	inputs0[4]	outputs[4]
30	N/A	None	8.064 ns	inputs1[4]	outputs[4]
31	N/A	None	8.038 ns	inputs0[6]	outputs[6]
32	N/A	None	8.031 ns	inputs1[6]	outputs[6]
33	N/A	None	8.023 ns	selinputs[0]	outputs[1]
34	N/A	None	7.850 ns	inputs2[4]	outputs[4]
35	N/A	None	7.706 ns	inputs1[2]	outputs[2]
36	N/A	None	7.678 ns	inputs1[1]	outputs[1]
37	N/A	None	7.639 ns	inputs0[2]	outputs[2]
38	N/A	None	7.609 ns	inputs2[2]	outputs[2]
39	N/A	None	7.242 ns	inputs2[1]	outputs[1]
40	N/A	None	7.064 ns	inputs0[1]	outputs[1]

c) 结果分析及结论

每个引脚只间相互传递产生的延时各不相同，挑选其中 p2p 时间的最大值，为 selinputs[1]传递给 outputs[0]。为 10.407ns，故整体延时为 10.407ns。

- 调用类属实体实现的 8-1 多路复用器

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

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	8.805 ns	inputs[6]	output
2	N/A	None	8.275 ns	inputs[5]	output
3	N/A	None	8.246 ns	inputs[4]	output
4	N/A	None	7.957 ns	inputs[2]	output
5	N/A	None	7.950 ns	inputs[3]	output
6	N/A	None	7.883 ns	selects[2]	output
7	N/A	None	7.865 ns	inputs[0]	output
8	N/A	None	7.806 ns	selects[1]	output
9	N/A	None	7.779 ns	selects[0]	output
10	N/A	None	7.651 ns	inputs[7]	output
11	N/A	None	7.491 ns	inputs[1]	output

c) 结果分析及结论

每个引脚只间相互传递产生的延时各不相同，挑选其中 p2p 时间的最大值，为 inputs[6]传递给 output。为 8.805ns，故整体延时为 8.805ns。

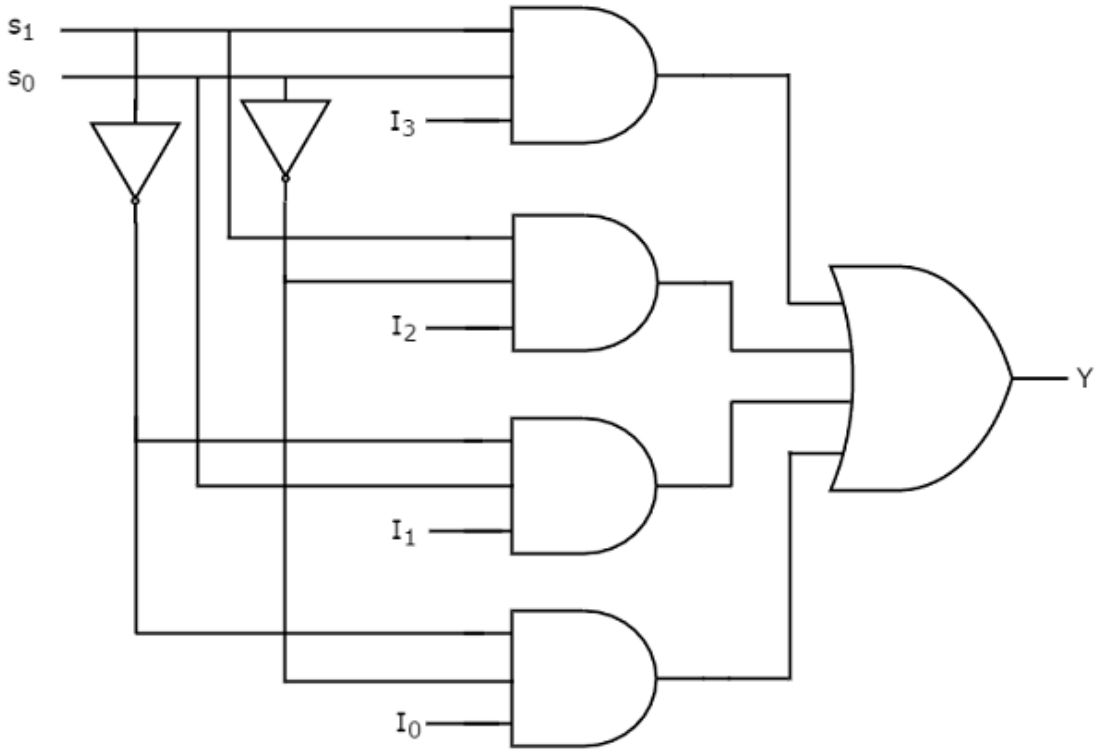
tpd (引脚到引脚的延时)

五、实验结论

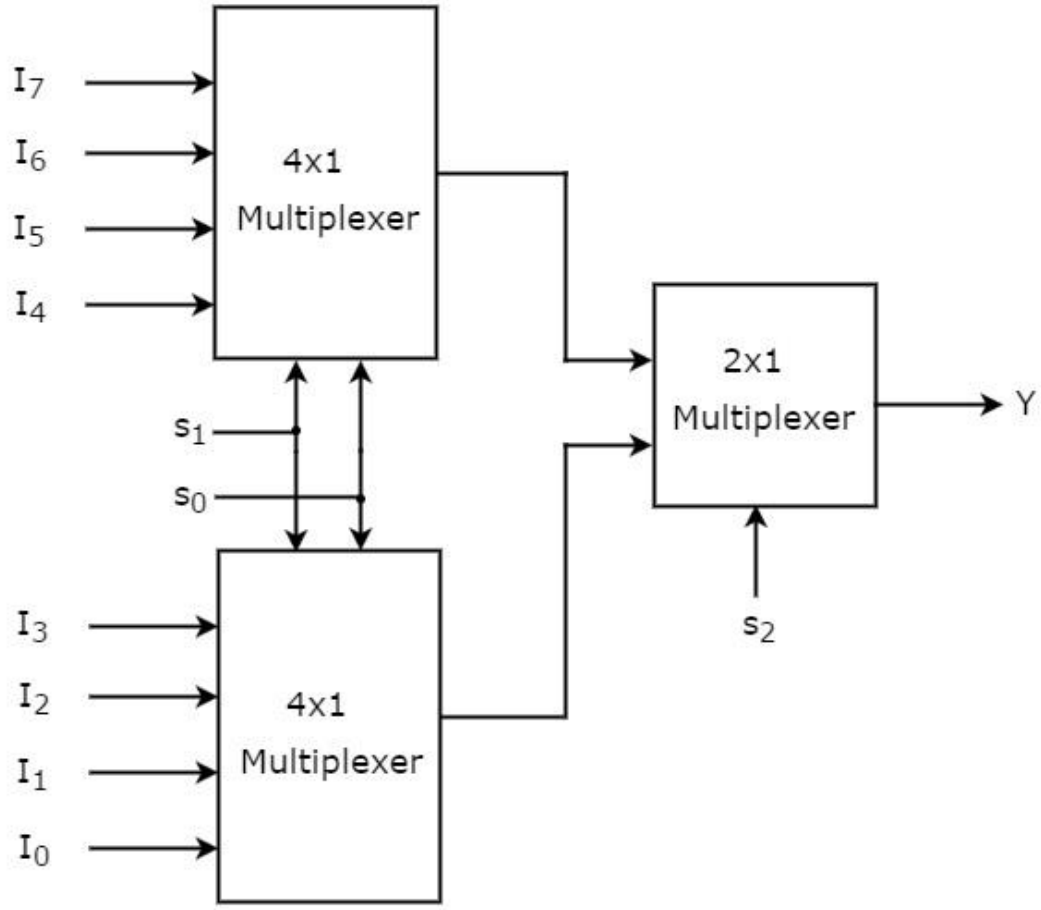
1、思考题

1） 经查阅相关资料，得到多路复用的以下实现方式

1. 使用译码器和与或门实现多路复用器，如图 4-2 多路复用器



2. 使用多层多路复用器串接成一个更大的多路复用器，如图 8-1 多路复用器



2) 用 VHDL 语言的描述方法有：

1. 顺序语句描述

   该种方法经常用到的语句有：

   - when-else 语句

2. if 选择语句描述

   该种方法经常用到的语句有：

   - if-then-else 语句

3. 并行语句描述

   该种方法经常用到的语句有：

   - 基本逻辑语句(and、not 等)

以上三种方法均需使用的语句有

库引入、实体声明、端口方向、结构体、库，程序包的调用，进程语句等

3、实验总结与实验心得

本次实验学习了用 VHDL 语言设计 8 重 3-1 多路复用器、设计 n-1 多路复用器，并调用该 n-1 多路复用器定制为 8-1 多路复用器、设计 4 位行波进位加法器、设计 4 位先行进位加法器。需要的能力有 VHDL 的简单编程能力，VHDL 的简单仿真能力，类属参数的设计能力，组合电路的设计与分析能力，实验报告的撰写能力，实验过程分析总结能力。通过使用软件实现硬件，既培养了编码能力，又增加了硬件设计能力和社会实践能力。该次实

验使我对组合电路的设计与 CPU 中命令的传递与选择有了更深刻的认识，巩固了组合电路的设计相关的知识点。我因此受益良多。