实验二 多路复用器与加法器的实现

班级 计科1907 姓名 杨杰 学号 201908010705

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、实验背景

1. 多路复用器，又名多路选择器、多路开关

多路复用器是一个组合电路，它可以从多个输入中选择一个输入，并将信息直接传输到输出。选择哪一条输入线由一组输入变量控制，它们被称为选择输入。通常，2^n条输入线要n个选择输入，选择输入的位组合决定选择哪个输入线。例如n=1的2-1多路复用器。这个复用器有两个信息输入I0和I1，一个单独的选择输入S，电路的真值表如表1所示。

表12-1多路复用器真值表

S	I_0	I_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

分析真值表可知，如果选择输入S=0，多路复用器输出为I0的值；如果选择输入S=1，多路开关输出I1的值。这样，S不是选择输入I0就是选择输入I1到输出Y。通过这些讨论，可以看出，2-1多路复用器输出Y的方程式为

$$Y=\overline{S}I_0+SI_1$$

2. 运用generic设计n位加法器，再调用n位加法器实现4位加法器。

a）n位加法器

```
libraryieee;
useieee.std_logic_1164.all;
useieee.std_logic_unsigned.all;

entitypar_addisgen
 eric(n:integer);
 port(a,b:instd_logic_vector(n-
        1downto0);s:outstd_logic_vector(n-
        1downto0);c:outstd_logic);
endpar_add;

architectureexaofpar_addis
signalt:std_logic_vector(ndownto0);b
egin
   t<=('0'&a)+('0'&b);
   s<=t(n-
   1downto0);c<=t(n);
endexa;
```

b）调用n位加法器，定制4位加法器

```
libraryieee;
useieee.std_logic_1164.all;
useieee.std_logic_unsigned.all;

entityadd3is
 port(x,y:instd_logic_vector(3downto0);ci
        n:instd_logic;
        s:outstd_logic_vector(3downto0);c
        out:outstd_logic);
endadd3;

architectureexaofadd3isco
mponentpar_addisgeneric
(n:integer);
 port(a,b:instd_logic_vector(n-
        1downto0);s:outstd_logic_vector(n-
        1downto0);c:outstd_logic);
endcomponent;
signalsum:std_logic_vector(3downto0);si
gnalmid:std_logic_vector(4downto0);sig
nalc3:std_logic;
begin
    g0:par_addgenericmap(n=>4)portmap(a=>x,b=>y,s=>sum,c=>c3);mid
    <=(c3&sum)+("0000"&cin);
```

```
        s<=mid(3downto0);c
        out<=mid(4);
    endexa;
```

3. 行波进位加法器

全加器

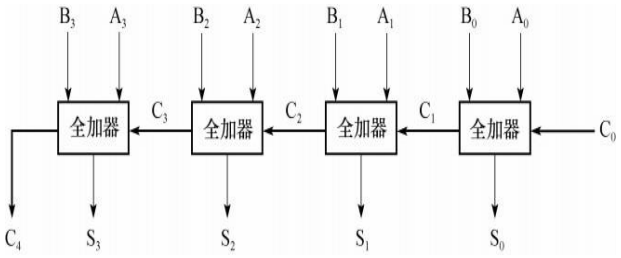
全加器是实现三位数相加的组合逻辑电路，共有三个输入，两个输出。输入变量中的两个用X和Y表示，代表两个加数，第三个输入Z表示低位产生的进位。两个输出用S(和)与C(进位)来表示。输出值由三位输入的算术和决定。当所有输入都为0时，输出均为0。当输入仅有一个为1或三个全为1时，输出S为1。当输入有两个或三个为1时，则输出C为1。全加器的布尔表达式用异或运算表示可写成：

$S=X\oplus Y\oplus Z$

$C=XY+YZ+XZ$

四位行波进位加法器

四位行波进位加法器由四个全加器级联形成。被加数A和加数B的下标从右至左依次递增，下标0表示最低有效位，进位位将整个全加器链式地连接起来。并行加法器的进位输入为C_0，进位输出为C_4。一个n位的行波进位加法器需要n个全加器,每个进位输出连接到下一个高位全加器的进位输入。



4. 先行进位加法器

行波进位加法器需要一级一级的进位，进位延迟很大。先行进位加法器（也叫超前进位加法器）可以有效的减少进位延迟。

设二进制加法器的第**i**位输入为X_i、Y_i,输出为S_i,进位输入为C_i，进位输出为C_{i+1}

则有

$$Si=Xi\oplus Yi\oplus Ci$$

$$Ci+1=Xi\cdot Yi+Xi\cdot Ci+Yi\cdot Ci=Xi\cdot Yi+(Xi+Yi)\cdot Ci$$

令Gi=Xi・Yi, Pi=Xi+Yi

则Ci+1=Gi+Pi・Ci

当Xi和Yi都为1时，Gi=1，产生进位C_{i+1}=1

当Xi和Yi有一个为1时，Pi=1，传递进位C_{i+1}=Ci

因此Gi定义为进位产生信号，Pi定义为进位传递信号。Gi的优先级比Pi高，也就是说：当Gi=1时（当然此时也有Pi=1），无条件产生进位，而不管Ci是多少;

当Gi=0而Pi=1时，进位输出为Ci,跟Ci之前的逻辑有关。

下面推导4位超前进位加法器。设4位加数和被加数为A和B，进位输入为Cin, 进位输出为Cout, 对于第i位的进位产生Gi=Ai·Bi, 进位传递Pi=Ai+Bi,i=0,1,2,3

这各级进位输出，递归的展开Ci，有：

C0=Cin

C1=G0+P0·C0

C2=G1+P1·C1=G1+P1·（G0+P0·C0）=G1+P1·G0+P1·P0·C0

C3=G2+P2·C2=G2+P2·G1+P2·P1·G0+P2·P1·P0·C0

C4=G3+P3·C3=G3+P3·G2+P3·P2·G1+P3·P2·P1·G0+P3·P2·P1·P0·C0

Cout=C4

由此可以看出，各级的进位彼此独立产生，只与输入数据和Cin有关，将各级间的进位级联传播给去掉了，因此减小了进位产生的延迟。

每个等式与只有三级延迟的电路对应，第一级延迟对应进位产生信号和进位传递信号，后两级延迟对应上面的积之和，这可从RTL视图中看到。

同时由真值表可以简单地得出第i位的和为：

Si=Xi⊕Yi⊕Ci=(Xi·Yi)⊕(Xi+Yi)⊕Ci=Gi⊕Pi⊕Ci

三、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。

四、实验要求

1. 进实验室前，请写一份预习报告；如有疑问，可在学习通平台相互讨论。

2. 预习报告内容有:

- 8重3-1多路复用器的VHDL程序；
- n-1多路复用器，调用该n-1多路复用器定制为8-1多路复用器的VHDL程序；
- 4位行波进位加法器的VHDL程序。
- 4位先行进位加法器的VHDL程序。

3. 在文本编辑区使用VHDL硬件描述语言设计功能块，再利用波形编辑区进行仿真，以此验证电路的逻辑功能是否正确，最后在Tool下用netlistviewer查看RTLviewer，以查看实现的RTL电路图。

4. 实验结束前，由指导老师检查了仿真波形后方可离开。

5. 最后撰写实验报告，提交到学习通平台，并在平台上分享设计的警告、资源消耗以及RTL视图。

6. 评判各种实现方案，并打分。

五、实验方法与步骤

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QuartusII。

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】-【newprojectwizard】-【next】-【next】（-【properties】（type=AHDL）-【next】（family=FLEX10K；name=EPF10K10TI144-4）-【next】-【finish】

(2).新建：【file】-【new】（第二个AHDLFile）-【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】-【startcompilation】进行文件编译，编译成功。

4、波形仿真及验证。新建一个vectorwaveformfile。

5、时序仿真或功能仿真。

6、查看RTLViewer:【Tools】-【netlistviewer】-【RTLviewer】。

六、实验过程

（1）8 重 3-1 多路复用器

1、编译过程

a）源代码如图（VHDL设计）

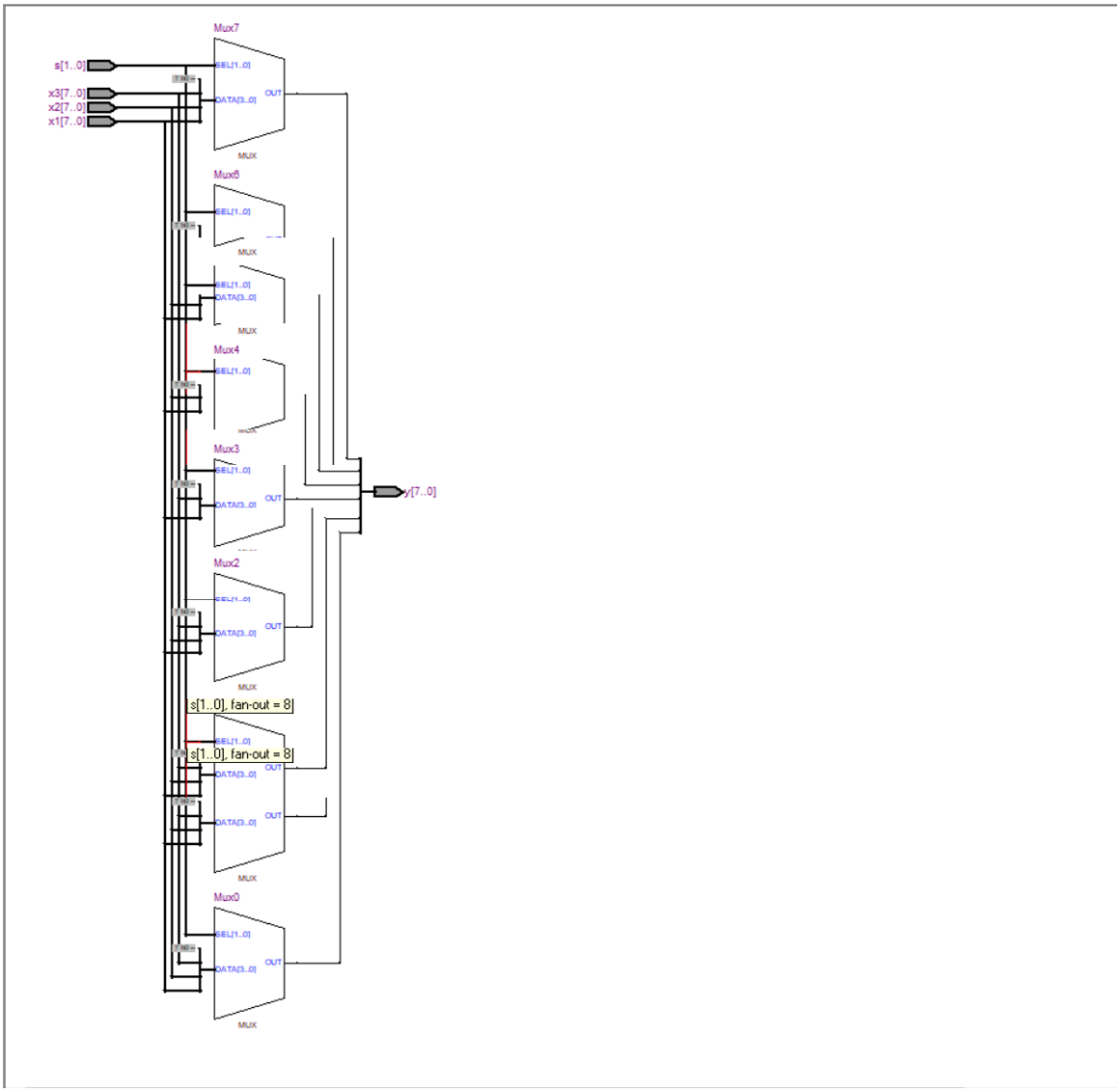
```
library ieee;
use ieee.std_logic_1164.all;
entity mul_8_3_1 is
port(s:in std_logic_vector (1 downto 0);
x1: in std_logic_vector(7 downto 0);
x2: in std_logic_vector(7 downto 0);
x3: in std_logic_vector(7 downto 0);
y:out std_logic_vector(7 downto 0));
end mul_8_3_1;
architecture one of mul_8_3_1 is
begin
with s select
y(0)<=x1(0) when"00",
      x2(0) when"01",
      x3(0) when"10",
      null when "11";
with s select
y(1)<=x1(1) when"00",
      x2(1) when"01",
      x3(1) when"10",
     null when "11";
with s select
y(2)<=x1(2) when"00",
      x2(2) when"01",
      x3(2) when"10",
      null when "11";
with s select
y(3)<=x1(3) when"00",
      x2(3) when"01",
```

```
30          x3(3) when"10",
31          null when "11";
32      with s select
33      y(4)<=x1(4) when"00",
34          x2(4) when"01",
35          x3(4) when"10",
36          null when "11";
37      with s select
38      y(5)<=x1(5) when"00",
39          x2(5) when"01",
40          x3(5) when"10",
41          null when "11";
42      with s select
43      y(6)<=x1(6) when"00",
44          x2(6) when"01",
45          x3(6) when"10",
46          null when "11";
47      with s select
48      y(7)<=x1(7) when"00",
49          x2(7) when"01",
50          x3(7) when"10",
51          null when "11";
52      end one;
```

b)编译、调试过程

编译通过，无警告信息。

c)RTL视图



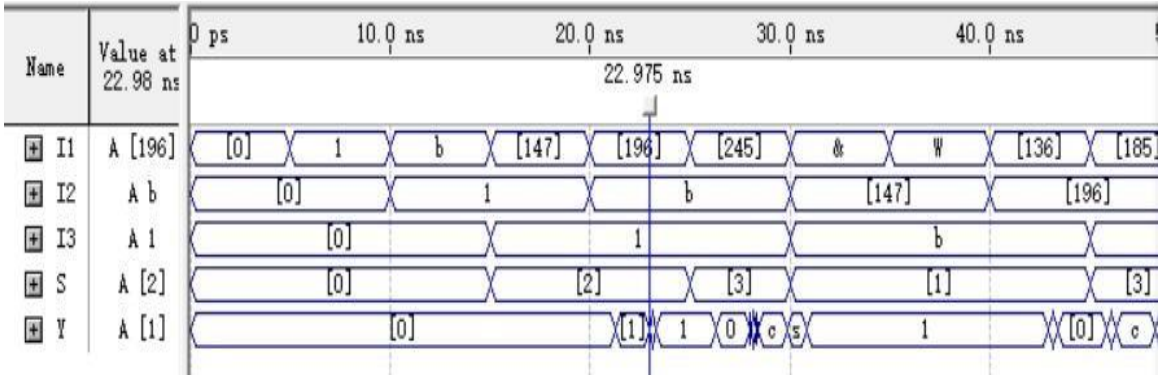
d)结果分析及结论

能够实现老师的全部要求。

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

当选择输入为 0 时，输出 I1;当选择输入为 2 时，输出为 I3；当选择输入为 1 时，输出为 I2；当输入 11 时，输出“XXXXXXXX”.结果正确，有延迟存在。

3、时序仿真

a)时序仿真过程

做好上述步骤后，编译【classictiminganalysis】-在compilationreport中选择【timinganalysis】-【tpd】（引脚到引脚的延时）

b)时序仿真图

tpd

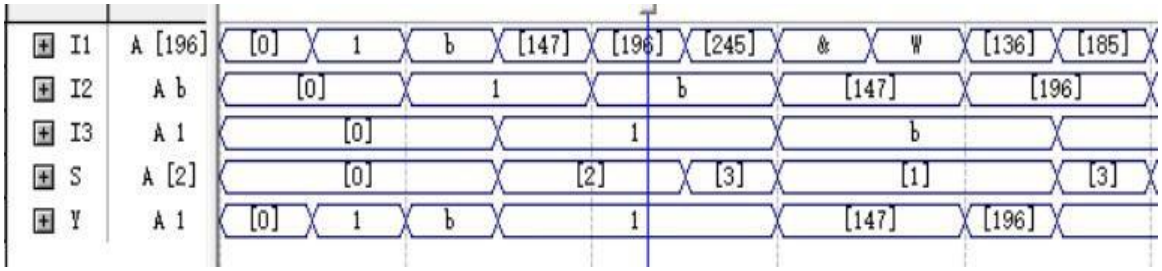
	Slack	Required P2P Time	Actual P2P Time	From	To
17	N/A	None	16.300 ns	S[0]	Y[5]
18	N/A	None	16.300 ns	S[0]	Y[4]
19	N/A	None	16.300 ns	S[0]	Y[3]
20	N/A	None	16.300 ns	S[0]	Y[2]
21	N/A	None	16.300 ns	S[0]	Y[1]
22	N/A	None	16.300 ns	I2[1]	Y[1]
23	N/A	None	16.300 ns	I2[0]	Y[0]
24	N/A	None	16.300 ns	I1[0]	Y[0]
25	N/A	None	16.300 ns	S[0]	Y[0]
26	N/A	None	15.800 ns	S[1]	Y[6]
27	N/A	None	15.800 ns	S[1]	Y[5]
28	N/A	None	15.800 ns	S[1]	Y[4]
29	N/A	None	15.800 ns	S[1]	Y[3]
30	N/A	None	15.800 ns	S[1]	Y[2]
31	N/A	None	15.800 ns	S[1]	Y[1]
32	N/A	None	15.800 ns	S[1]	Y[0]
33	N/A	None	15.400 ns	I3[7]	Y[7]
34	N/A	None	14.900 ns	I3[5]	Y[5]
35	N/A	None	14.900 ns	I3[4]	Y[4]
36	N/A	None	14.800 ns	I3[3]	Y[3]
37	N/A	None	14.800 ns	I3[2]	Y[2]
38	N/A	None	14.700 ns	I3[1]	Y[1]
39	N/A	None	14.600 ns	I3[6]	Y[6]
40	N/A	None	12.900 ns	I3[0]	Y[0]

4、功能仿真

a)功能仿真过程

做好上述步骤后，在【processing】中选择【simulatortool】-【generatefunctionalsimulationnetlist】-【startsimulation】

b)功能仿真图



c)结果分析及结论

功能选择没有延迟，当选择输入为0时，输出I1;当选择输入为2时，输出为I3；当选择输入为1时，输出为I2；当输入11时，输出“XXXXXXXX”.结果正确。

（2）调用 n-1 多路复用器定制为 8-1 多路复用器

1.编译过程

a）源代码如图（VHDL设计）

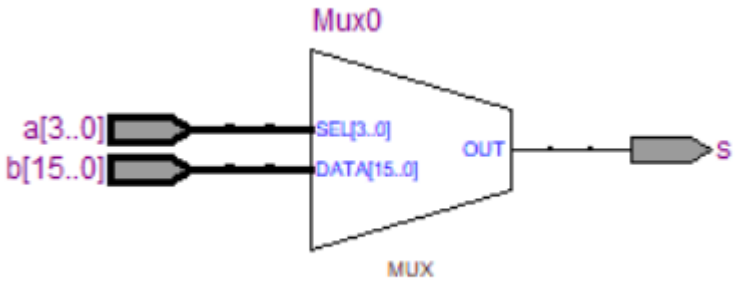
```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.std_logic_unsigned.all;
 4  entity TSOC is
 5  generic(n:integer:=4);
 6  port(a:in std_logic_vector(n-1 downto 0);
 7       b:in std_logic_vector(2**n-1 downto 0);
 8       s:out std_logic
 9       );
10  end TSOC;
11  architecture st of TSOC is
12  begin
13  process(a,b)
14  variable i:integer range 0 to 2**n-1;
15  begin
16  i:=conv_integer(a);
17  s<=b(i);
18  end process;
19  end st;
```

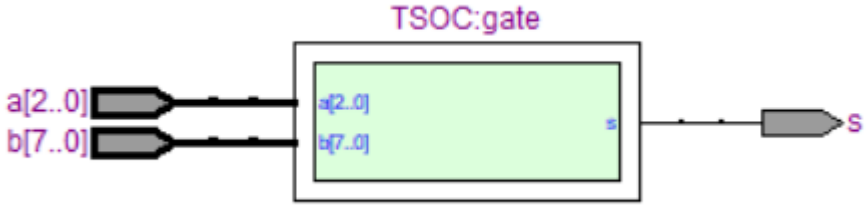
```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4   entity n_1_mul is
5   port(a:in std_logic_vector(2 downto 0);
6        b:in std_logic_vector(7 downto 0);
7        s:out std_logic
8        );
9   end n_1_mul;
10  architecture s of n_1_mul is
11  component TSOC is
12  generic (n:integer:=3);
13  port(a:in std_logic_vector(n-1 downto 0);
14       b:in std_logic_vector(2**n-1 downto 0);
15       s:out std_logic
16       );
17  end component;
18  begin
19  gate:TSOC generic map(3) port map(a,b,s);
20  end s;
```

b)编译、调试过程

编译通过，无警告信息。

c)RTL视图





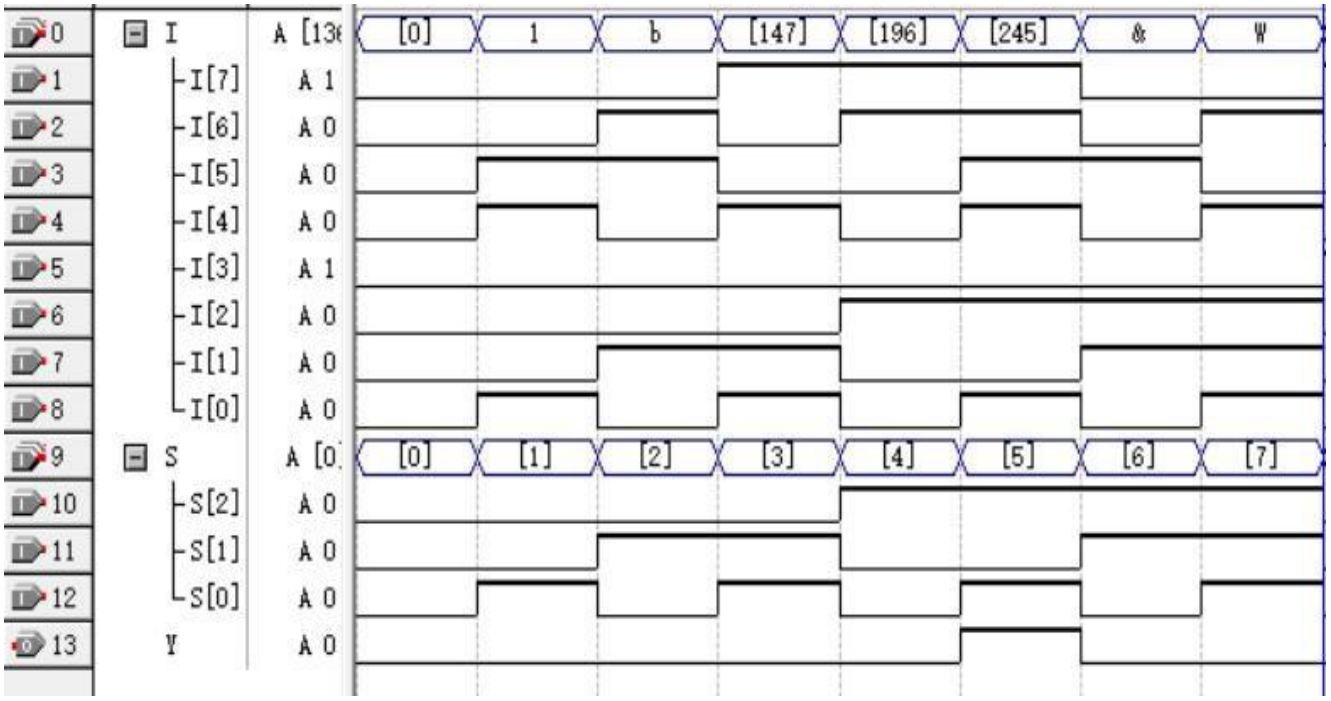
d)结果分析及结论

能够实现老师的全部要求。

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

当输入为000时，输出I0；当输入为001时，输出I1；当输入为010时，输出I2；当输入为011时，输出I3；当输入为100时，输出I4；当输入为101时，输出I5；当输入为110时，输出I6；当输入为111时，输出I7;结果正确，存在延迟。

3.时序仿真

a)时序仿真过程

做好上述步骤后，编译【classictiminganalysis】-在compilationreport中选择【timinganalysis】-【tpd】（引脚到引脚的延时）

b)时序仿真图

tpd

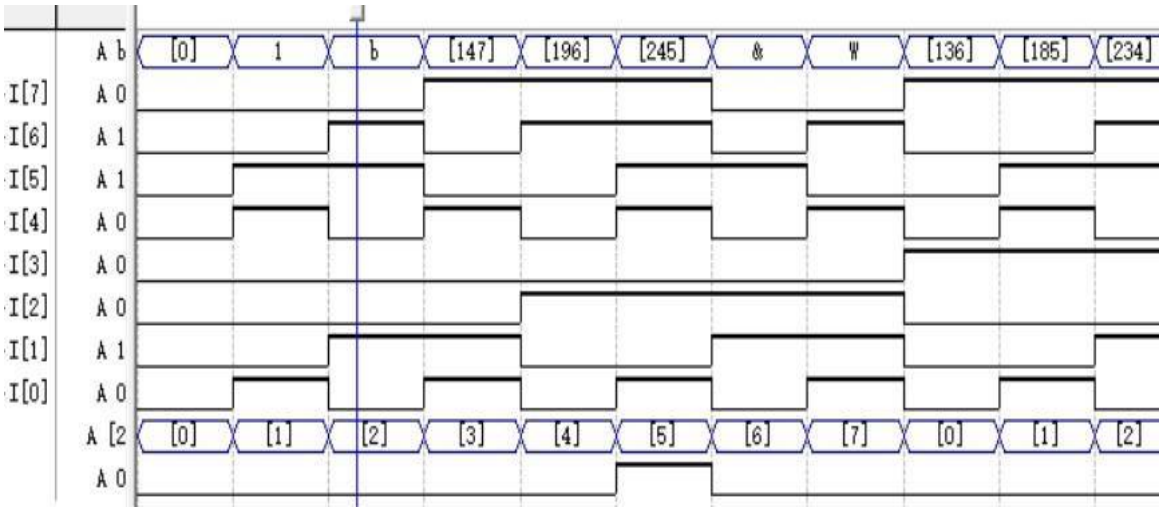
	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	20.900 ns	I[1]	Y
2	N/A	None	20.700 ns	I[0]	Y
3	N/A	None	19.100 ns	S[0]	Y
4	N/A	None	19.100 ns	I[6]	Y
5	N/A	None	19.100 ns	S[1]	Y
6	N/A	None	18.600 ns	I[4]	Y
7	N/A	None	18.300 ns	I[2]	Y
8	N/A	None	17.800 ns	I[3]	Y
9	N/A	None	16.200 ns	I[5]	Y
10	N/A	None	15.700 ns	I[7]	Y
11	N/A	None	14.600 ns	S[2]	Y

4.功能仿真

a)功能仿真过程

做好上述步骤后，在【processing】中选择【simulatortool】-【generatefunctionalsimulationnetlist】-【startsimulation】

b)功能仿真图



c)结果分析及结论

当输入为000时，输出I0；当输入为001时，输出I1；当输入为010时，输出I2；当输入为011时，输出I3；当输入为100时，输出I4；当输入为101时，输出I5；当输入为110时，输出I6；当输入为111时，输出I7;结果正确，不存在延迟。

七、实验心得

收获：

1）在实验的具体操作中，对多路复用器以及加法器有了更近一步的了解。

2）对行波加法器和先行进位加法器有了更深入的理解。

3）学习了如何利用component原件例化语句以及port map的方式实现多个模块之间信号的互联。

4）学习了如何利用generic类属参数语句设计实体和其外部环境通信的参数传递静态的信息，使可以在实体之外改变参数n的值，从而改变电路的规模。

实验操作应注意：

（1）顶层文件的实体名只能有一个，而且注意符号文件不能与顶层文件的实体名相同。

（2）保存波形文件时，注意文件名必须与工程名一致，因为在多次为一个工程建立波形文件时，一定要注意保存时文件名要与工程名一致，否则不能得到正确的仿真结果。

八、思考题

1. **多路复用器的实现方法很多，请总结两种以上实现方法。**

   a) when-else的VHDL数据流描述法

   b）with-select的VHDL数据流描述法

   c）二进制转化与generic的参数化VHDL描述法

   d）Verilog结构描述法

   e）采用布尔表达式的Verilog条件数据流描述法

2. **总结VHDL语言描述多路复用器的方法和常用语句。**

   方法：

   第一种：在实体中定义选择输入、输入、输出信号，在结构体中按照要求利用when-else语句进行选择输出。

   第二种：在顶层模块利用generic设计一个n-1多路复用器，在底层模块给n赋值，两个模块之间的信号通过portmap的方式，实现模块之间的信号互联。

   常用语句：

   并行语句、when-else语句

   generic类属参数语句：GENERIC([常数名 数据类型 [: 设定值]

   条件信号赋值语句：

   语法结构：

目标信号<=表达式1 when 条件1 else

表达式2 when 条件2 else

表达式3 when 条件3 else

…

表达式n-1 when 条件n-1 else

表达式n；

选择信号赋值语句：

语法结构：

with 表达式 select

目标信号<=表达式1 when 选择条件1，

表达式2 when 选择条件2，

…

表达式n when 选择条件n；

component元件例化语句：

语法格式：

component引用的元件名

generic参数说明；

port端口说明；

endcomponet；