

数字电路与逻辑设计

课程实验报告

|  |  |
| --- | --- |
| 题目： | 加法器、运算器的设计 |
| 学生姓名： | 魏子铖 |
| 学生学号： | 201726010308 |
| 专业班级： | 软件1703 |
| 完成时间： | 2018.12.6 |

**实验三 加法器、运算器的设计**

班级 软件1703 姓名 魏子铖 学号 201706010308

1. **实验目的**

1．用VHDL语言设计全加器；

2．利用设计的全加器组成四位并行加法器；

3．用逻辑图和VHDL语言设计四位串行加法器。

4．使用VHDL语言设计模型机运算器ALU。

1. **实验内容**

1．熟悉Quartus II软件的基本操作，了解各种设计方法（原理图设计、文本设计、波形设计）

2．用逻辑图和VHDL语言设计一个全加器，最后仿真验证。

3．利用设计的全加器，用逻辑图和VHDL语言设计一个四位并行加法器，最后仿真验证。

4．用逻辑图和VHDL语言设计一个四位串行加法器，最后仿真验证。

5．使用VHDL语言设计模型机运算器ALU，最后仿真验证。

1. **实验方法**
2. **实验方法**

**用VHDL语言设计一个全加器**

1. 新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为FullAdder\_VHDL +设置top-level design entity name为FullAdder\_VHDL

\_VHDL）-【finish】

(2).新建：【file】-【new】-【VHDL File】-【OK】

2.写好源代码，保存文件（FullAdder\_VHDL.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4个警告，文件编译成功。

4.波形仿真及验证。

(1).新建一个vector waveform file, 保存文件（FullAdder\_VHDL.vwf）。。

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为8ns）-【ok】）。

(4).按照程序所述插入X,Y,Z,C,S三个节点（X,Y,Z为输入节点，C,S为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

(5).设置X,Y,Z的输入波形…点击保存按钮保存。

（操作为：点击name（如：Z）-右击-【value】-【clock】（如设置period=2ns；offset=0ns），同理设置name X,Y（如period=8/4ns；offset=0ns），保存）。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name C，S的输出图。

5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

**用逻辑图设计一个全加器**

1.新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为FullAdder +设置top-level design entity name为FullAdder）-【finish】

(2).新建：【file】-【new】-【Block Diagrame/Schematic File】-【OK】

2.画好原理图，保存文件（FullAdder.bdf）

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5个警告，文件编译成功。

4.波形仿真及验证。

(1).新建一个vector waveform file, 保存文件（FullAdder\_VHDL.vwf）。。

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为8ns）-【ok】）。

(4).按照程序所述插入X,Y,Z,C,S三个节点（X,Y,Z为输入节点，C,S为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

(5).设置X,Y,Z的输入波形…点击保存按钮保存。

（操作为：点击name（如：Z）-右击-【value】-【clock】（如设置period=2ns；offset=0ns），同理设置name X,Y（如period=8/4ns；offset=0ns），保存）。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name C，S的输出图。

5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

**用VHDL语言，利用设计的全加器组成四位并行加法器**

1.新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为Adder\_4\_VHDL+设置top-level design entity name为Adder\_4\_VHDL）-【finish】

(2).新建：【file】-【new】-【VHDL File】-【OK】

2. 利用写好的FullAdd\_VHDL.vhd文件，添加到顶层文件中作为元件。写好源代码，保存文件（Adder\_4\_VHDL.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5个警告，文件编译成功。

4.波形仿真及验证。

(1).新建一个vector waveform file，保存文件（Adder\_4\_VHDL.vwf）。

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为512ns）-【ok】）。

(4).按照程序所述插入C0，A[3..0], B[3..0]，C4,S[3..0]（C0，A[3..0], B[3..0]为输入节点，C4,S[3..0]为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

(5).设置C0，A[3..0], B[3..0]的输入波形…点击保存按钮保存。

（操作为：点击name（如：B[0]））-右击-【value】-【clock】（如设置period=2ns；offset=0ns），同理设置name B[1..3]A[0..3]（如period=4/8/16/32/64/128/256ns；offset=0ns），name C0（设置为‘0’）,保存）。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name C4和S[3..0]的输出图。

5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

**用逻辑图，利用设计的全加器组成四位并行加法器**

1.新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为Adder\_4\_VHDL +设置top-level design entity name为Adder\_4\_VHDL）-【finish】

(2).新建：【file】-【new】-【Block Diagrame/Schematic File】-【OK】

2.利用写好的FullAdd.bdf文件，保存为FullAdd.bsf文件，添加到顶层文件中作为元件。写好源代码，保存文件（Adder\_4\_VHDL.bdf）

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5个警告，文件编译成功。

4.波形仿真及验证。

(1).新建一个vector waveform file，保存文件（Adder\_4\_VHDL.vwf）。

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为512ns）-【ok】）。

(4).按照程序所述插入C0，A[3..0], B[3..0]，C4,S[3..0]（C0，A[3..0], B[3..0]为输入节点，C4,S[3..0]为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

(5).设置C0，A[3..0], B[3..0]的输入波形…点击保存按钮保存。

（操作为：点击name（如：B[0]））-右击-【value】-【clock】（如设置period=2ns；offset=0ns），同理设置name B[1..3]A[0..3]（如period=4/8/16/32/64/128/256ns；offset=0ns），name C0（设置为‘0’）,保存）。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name C4和S[3..0]的输出图。

5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

**用VHDL语言设计4位串行加法器**

1.新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为SerialAdder\_4\_VHDL+设置top-level design entity name为SerialAdder\_4\_VHDL）-【finish】

(2).新建：【file】-【new】-【VHDL File】-【OK】

2.写好源代码，保存文件（SerialAdder\_4\_VHDL.vhd和FullAdder.vhd和SRG\_4.vhd和Dffx.vhd）

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5个警告，文件编译成功。

4、波形仿真及验证。

(1).新建一个vector waveform file,保存文件(SerialAdder\_4\_VHDL.vwf)

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为160us）-【ok】）。

(4).按照程序所述插入Shift,Clock,Reset,SI为输入节点，Q,O为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

(5).设置Shift,Clock,Reset,SI的输入波形…点击保存按钮保存。

（操作为：点击name（如：Reset））-在波形图中选取合适的一段设定,值为0，之后设置为1，同理设置name Shift(全部为1)Clock(在Reset后连续12个周期)的各个节点，SI（可以任意设置），保存）。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name Q，O的输出图。

5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

**用逻辑图设计4位串行加法器**

1.新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为SerialAdder\_4+设置top-level design entity name为SerialAdder\_4）-【finish】

(2).新建：【file】-【new】-【VHDL File】-【OK】

2. 利用写好的FullAdd.bdf文件，保存为FullAdd.bsf文件，添加到顶层文件中作为元件。同理，编写SRG4\_CLR（带重置功能的SRG4寄存器）保存为SRG4\_CLR.bsf文件，添加到顶层文件中作为元件写好源代码，保存文件（SerialAdder\_4.bdf）

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25个警告，文件编译成功。

4、波形仿真及验证。

(1).新建一个vector waveform file,保存文件(SerialAdder\_4.vwf)

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为150us）-【ok】）。

(4).按照程序所述插入Shift,Clock,Reset,SI为输入节点，Q,O为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

(5).设置Input[7..0]的输入波形…点击保存按钮保存。

（操作为：点击name（如：Reset））-在波形图中选取合适的一段设定,值为0，之后设置为1，同理设置name Shift(全部为1)Clock(在Reset后连续12个周期)的各个节点，SI（可以任意设置），保存）。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name Q，O的输出图。

5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

**用VHDL语言设计模型机运算器ALU**

1.新建，编写源代码。

(1).选择保存项：【File】-【new project wizard】-【next】（设置文件路径+设置project name为ALU+设置top-level design entity name为ALU）-【finish】

(2).新建：【file】-【new】-【VHDL File】-【OK】

2.写好源代码，保存文件（ALU.vhd）

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5个警告，文件编译成功。

4、波形仿真及验证。

(1).新建一个vector waveform file,保存文件(ALU.vwf)

(2).设置仿真模式【Assignments】-【Setting】-【Simulator Settings】-【Simulation mode】选择Functional

(3).设置end time（操作为：【edit】-【End Time】（设置为160us）-【ok】）。

(4).按照程序所述插入Input[7..0]为输入节点，Output[1..0]为输出节点）。

(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。

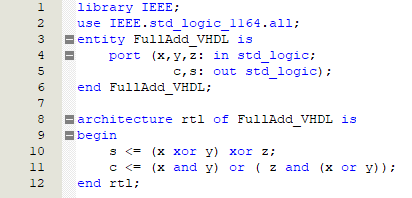
(5).设置Input的输入波形（按照需要测试的数据进行设置，例如前四位一次设置为“1001” “0110”“1011”“0101”“1010”,后四位随机测试）,点击保存按钮保存。

(6).然后【Processing】-【Generate Functional Simulation Netlist】;

(7).然后【Processing】-【start simulation】，形成name Output[1..0]的输出图。

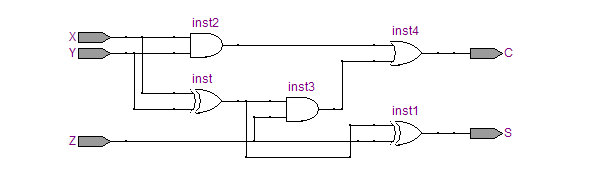
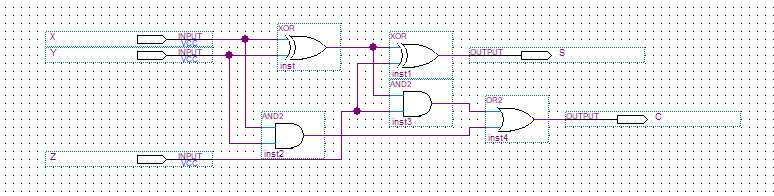
5.查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

1. **实验过程**
2. **a．用VHDL语言和逻辑图设计一个全加器**

**编译过程**

a）源代码如图（VHDL设计）

原理图如图

RTL如图

b)编译、调试过程

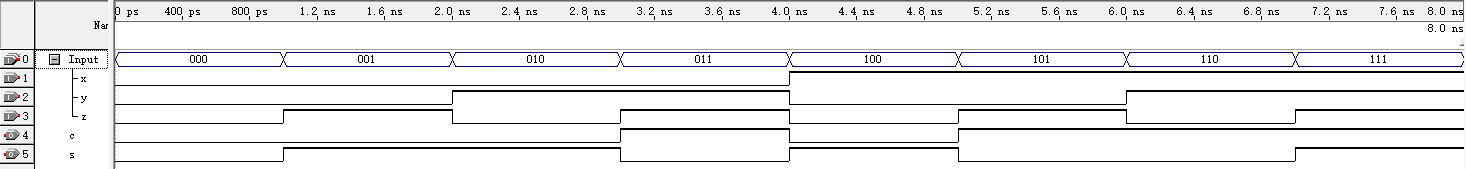
编译通过，有4个警告；

c)结果分析及结论

X,Y,Z端口输入3个二进制数，相加后最后一位结果通过S输出，进位通过C输出。

**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

0-1ns：X,Y,Z输入000，相加结果为0，S输出0，C输出0，正确

1-2ns：X,Y,Z输入001，相加结果为1，S输出1，C输出0，正确

2-3ns：X,Y,Z输入010，相加结果为1，S输出1，C输出0，正确

3-4ns：X,Y,Z输入011，相加结果为10，S输出0，C输出1，正确

4-5ns：X,Y,Z输入100，相加结果为1，S输出1，C输出0，正确

5-6ns：X,Y,Z输入101，相加结果为10，S输出0，C输出1，正确

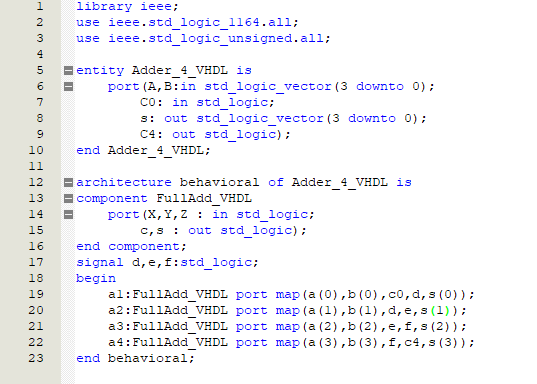
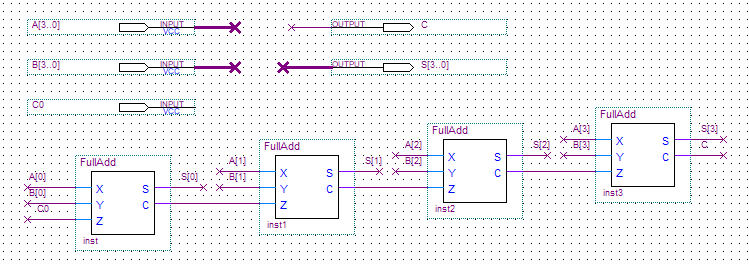
6-7ns：X,Y,Z输入110，相加结果为10，S输出0，C输出1，正确

7-6ns：X,Y,Z输入111，相加结果为11，S输出1，C输出1，正确

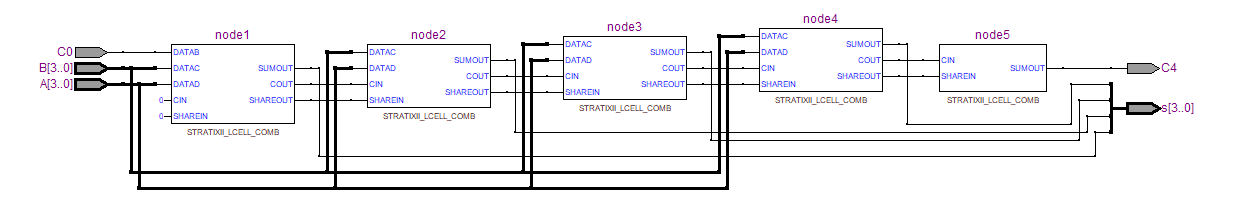
**b．利用设计的全加器，用VHDL语言和原理图设计四位并行加法器**

**编译过程**

a）源代码如图（VHDL设计）



逻辑图如图

RTL如图

b)编译、调试过程

编译通过，有5个警告；

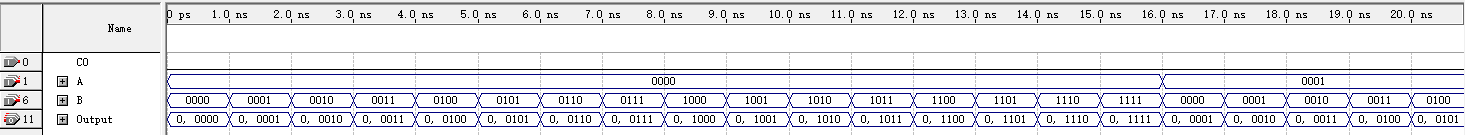
c)结果分析及结论

C0提供输入但是固定输入为0，其他4个全加器分别处理最低位到最高位，每一个全加器输出一位结果，最后一个全加器额外输出一个进位。

例如输入“0000”和“1111”，输出结果为‘0’和“1111”；

**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

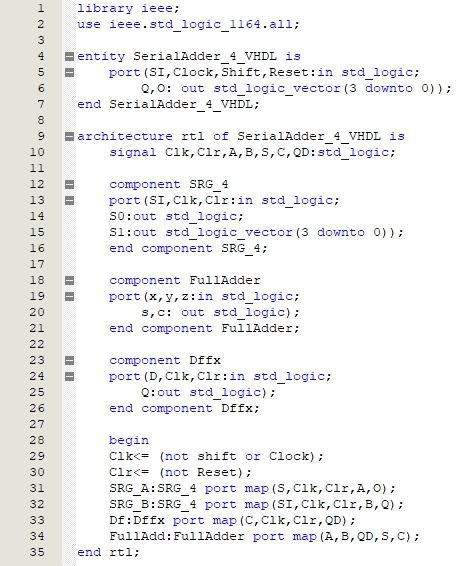
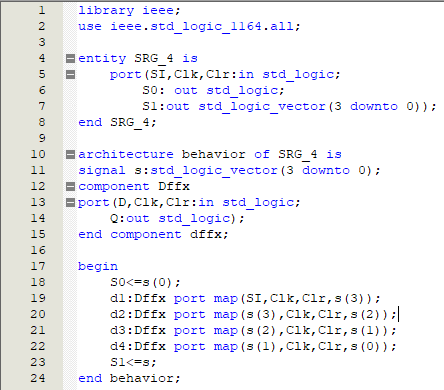
c)结果分析及结论

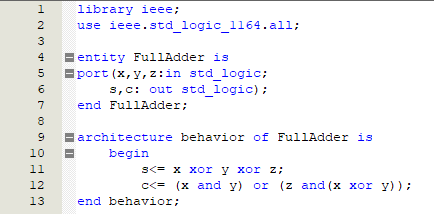
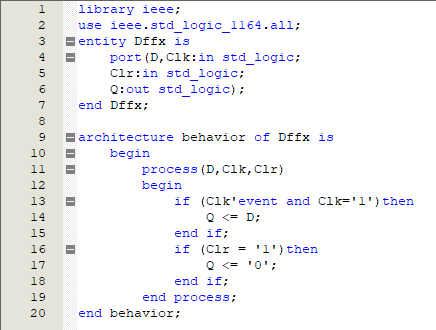
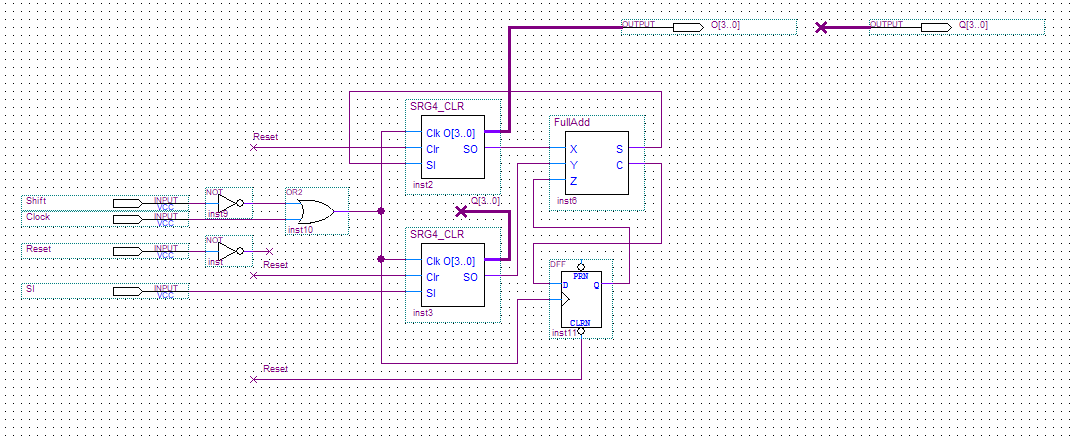
以2ns为一周期，每周期两个4位二进制数相加，得到得5位结果（包括最后一位进位和4位结果），均正确

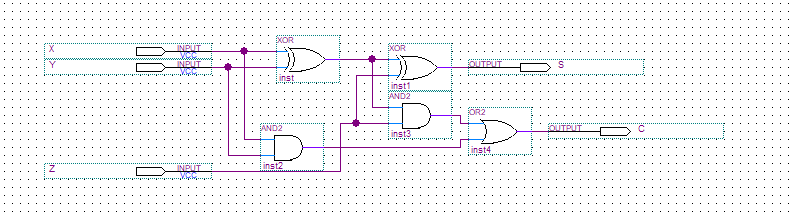
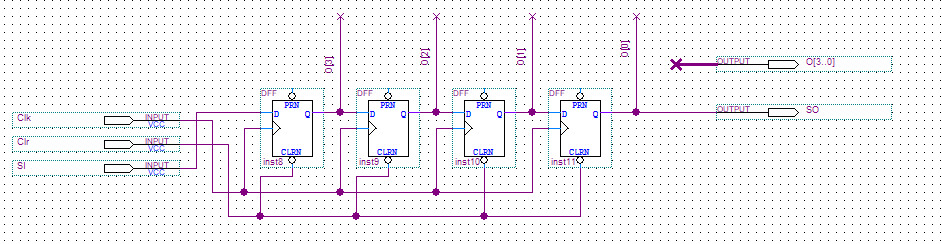
**c.用VHDL语言和原理图设计一个串行加法器**

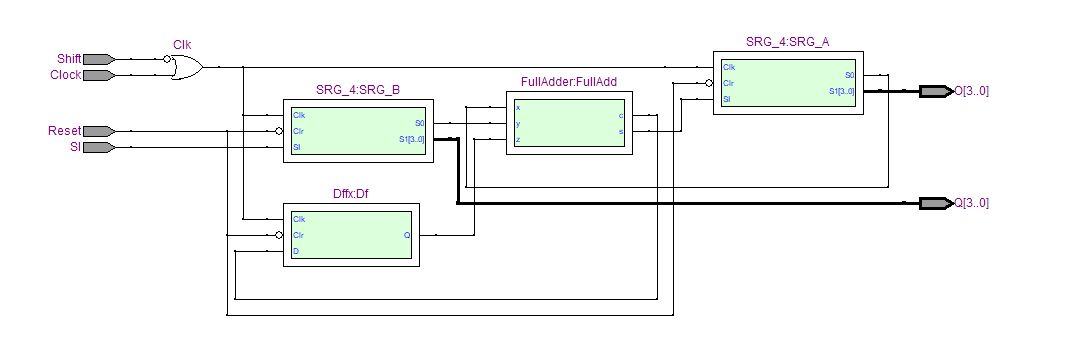
**编译过程**

a）源代码如图（VHDL设计）



原理图如图：

RTL如图



b)编译、调试过程

编译通过，有5个警告；

c)结果分析及结论

为了不出现未定义的情况，先进行一次重置，Reset在一开始设0，将所有的触发器触发成复位。然后连续12个时钟信号周期，同时Shift始终输入1，保证时钟信号有效。

在前4个周期的上升沿，依次从低到高输入4位二进制数，一位每周期存入B寄存器中；

在第5到第8个周期的上升沿，B寄存器的4位二进制数从低位到高位，一位每周期，通过全加器与‘0’相加后，原值存入到A寄存器中，同时从低到高输入新的4位二进制数，一位每周期存入B寄存器中；

在第9到第12个周期的上升沿，每一个周期将B寄存器的一位二进制数输出，与A寄存器的一位二进制数以及进位结果（第一次为0）相加，将不包括进位的结果重新存入A寄存器中，进位结果在下个周期与A，B寄存器输出的二进制数相加。

第12个周期的结果就是两个4位二进制数除去最高位进位的相加结果。

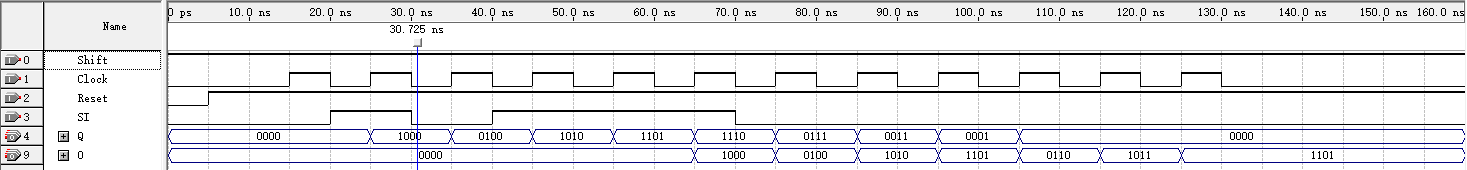
为了方便调试和观察结果，我们对两个寄存器的4个触发器保存的值进行输出，输出端口为O【3..0】和Q【3..0】

O【3..0】和Q【3..0】的结果应与上述原理逻辑一致。

**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

Shift始终设置为1，保证时钟信号有效

一开始Reset设置为0，对整个加法器进行重置。

在时钟信号的第一个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0000”，O不变，为“0000”；

在时钟信号的第二个周期，SI输入为1，B寄存器在上升沿保存该值，Q结果为“1000”，O不变，为“0000”；

在时钟信号的第三个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0100”，O不变，为“0000”；

在时钟信号的第四个周期，SI输入为1，B寄存器在上升沿保存该值，Q结果为“1010”，O不变，为“0000”；

在时钟信号的第五个周期，SI输入为1，B寄存器在上升沿保存该值，Q结果为“1101”，B的输出一位在A的输出一位与全加器相加后，保存到O的最高位，O变为“0000”；

在时钟信号的第六个周期，SI输入为1，B寄存器在上升沿保存该值，Q结果为“1110”，B的输出一位在A的输出一位与全加器相加后，保存到O的最高位，O变为“1000”；

在时钟信号的第七个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0111”，B的输出一位在A的输出一位与全加器相加后，保存到O的最高位，O变为“0100”；

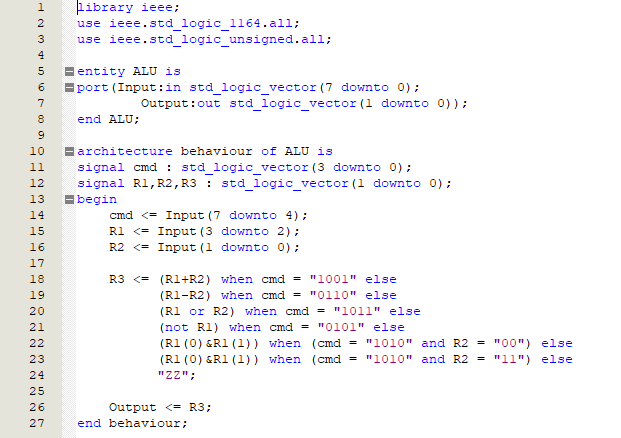
在时钟信号的第八个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0011”，B的输出一位在A的输出一位与全加器相加后，保存到O的最高位，O变为“1010”；

在时钟信号的第九个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0001”，B的输出一位在与全加器相加后，保存到O的最高位，O变为“1101”；

在时钟信号的第十个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0000”，B的输出一位在与全加器相加后，保存到O的最高位，O变为“0110”；

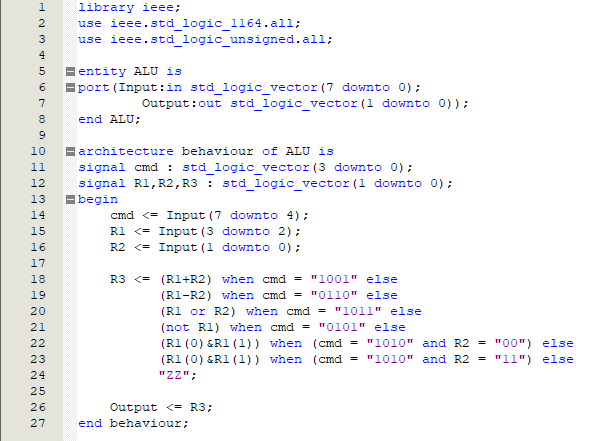
在时钟信号的第十一个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0000”，B的输出一位在与全加器相加后，保存到O的最高位，O变为“1011”；

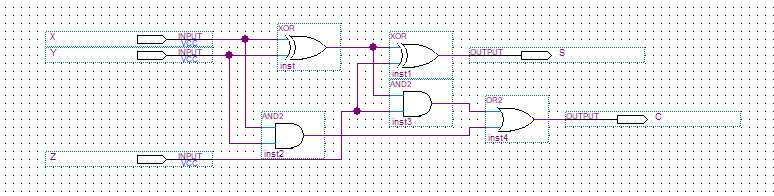
在时钟信号的第十二个周期，SI输入为0，B寄存器在上升沿保存该值，Q结果为“0000”，B的输出一位在与全加器相加后，保存到O的最高位，O变为“1101”；

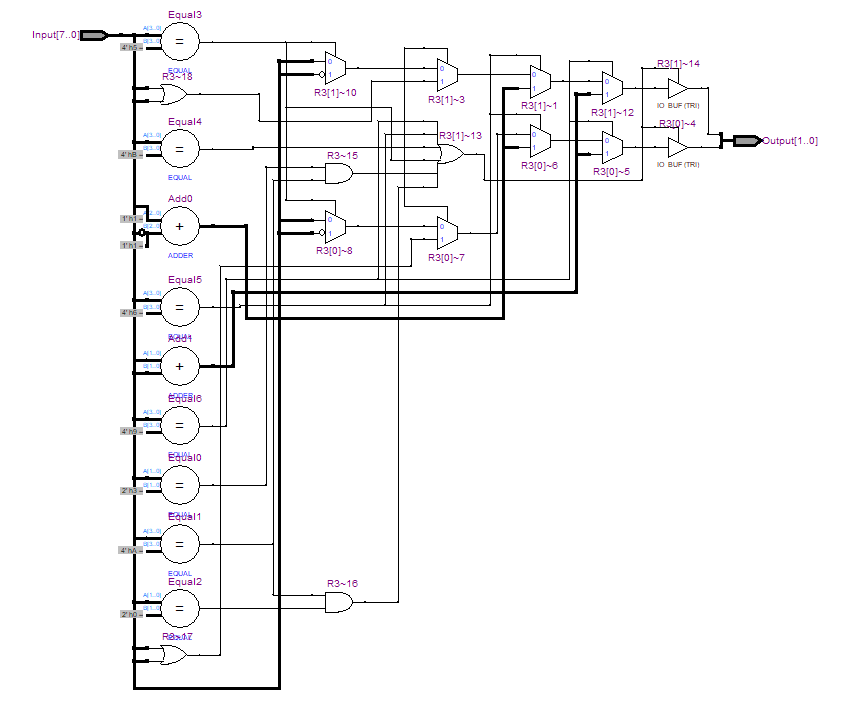
以上结果符合预期，正确

**d．用VHDL语言和逻辑图设计一个全加器**

**编译过程**

 a）源代码如图（VHDL设计）



RTL如图

b)编译、调试过程

编译通过，有4个警告；

c)结果分析及结论

当前四位为“1001”时，将后四位中的前两位与后两位相加，保存到R1中，输出；

当前四位为“0110”时，将后四位中的前两位与后两位相减，保存到R1中，输出；

当前四位为“1011”时，将后四位中的前两位与后两位做或运算，保存到R1中，输出；

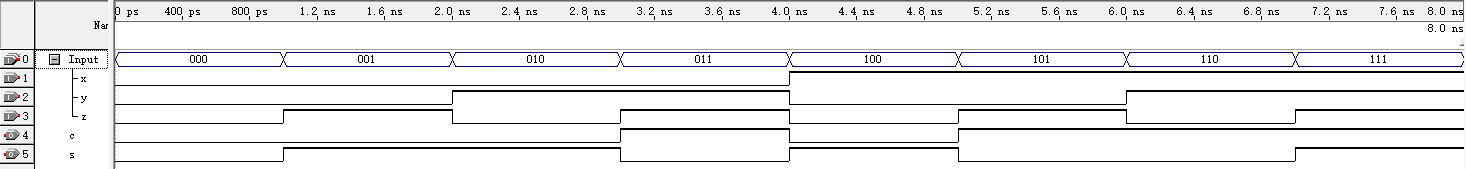
当前四位为“0101”时，将后四位中的前两位做非运算，保存到R1中，输出；

当前四位为“1010”，最后两位是“00”时，将后四位中的前两位做循环右移计算，保存到R1中，输出；

当前四位为“1001”时，最后两位是“11”时，将后四位中的前两位做循环左移计算，保存到R1中，输出；

**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

当前四位为“1001”时，将后四位中的前两位与后两位相加，保存到R1中，输出；

当前四位为“0110”时，将后四位中的前两位与后两位相减，保存到R1中，输出；

当前四位为“1011”时，将后四位中的前两位与后两位做或运算，保存到R1中，输出；

当前四位为“0101”时，将后四位中的前两位做非运算，保存到R1中，输出；

当前四位为“1010”，最后两位是“00”时，将后四位中的前两位做循环右移计算，保存到R1中，输出；

当前四位为“1001”时，最后两位是“11”时，将后四位中的前两位做循环左移计算，保存到R1中，输出；

仿真中结果均符合预期，正确

1. **实验结论**

本次实验内容多，难度较大，完成这次实验是一次很大的挑战。在做预习报告之前，对实验内容的很多概念都不是很了解，比如锁存器，触发器，寄存器。在做实验之前先认真的看了书本，也用quarters自己实现了一遍锁存器和触发器，让我对于书本知识和实验内容有了一个深刻的认识。本次实验内容很多要用到VHDL语音，经常查资料和问同学，再加上自己的尝试，最后也掌握了不少VHDL的语法。本次实验难度虽大，但也收获很多。