**基于fpga的数字系统设计 实验报告**

|  |  |  |
| --- | --- | --- |
| 评 语 | **成绩** |  |
| 教 师：  年 月 日 | | |

**学院班级： 171301201**

**学生学号： 17130120116**

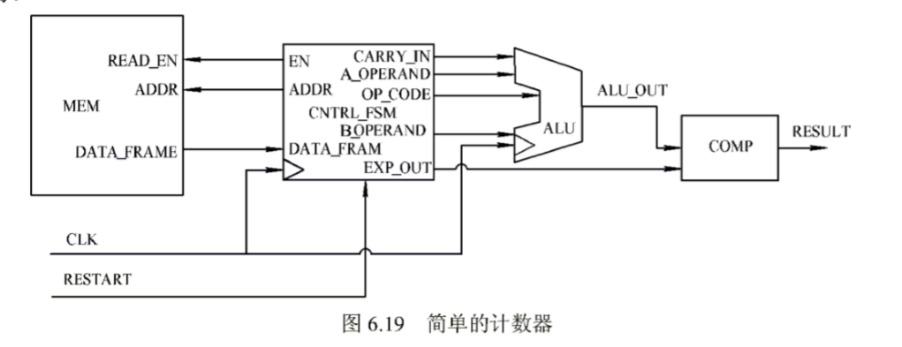
**学生姓名： 李云水**

**实验日期： 2020/5/23**

**实验题目： 算术逻辑单元实验**

**实验环境：联想小新潮7000 i7 8G ，win10，ISE Design Suite 14.7**

**一、实验介绍**

本次实验将完成图6.19中算术逻辑单元ALU的RTL描述。

**二、实验目标**

1.学习使用case语句描述ALU的结构；

2.验证逻辑结构；

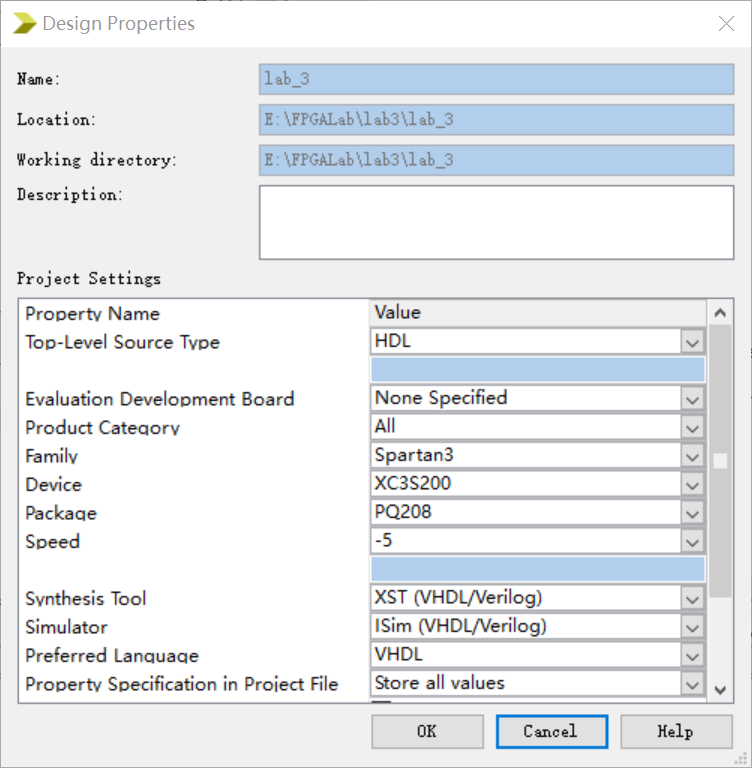
3.为计数器项目创建一个子模块；

4.为输出数据的比较做准备。

**三、实验过程**

**1.创建一个新的工程**

**配置属性**

****

**2.使用case描述语句**

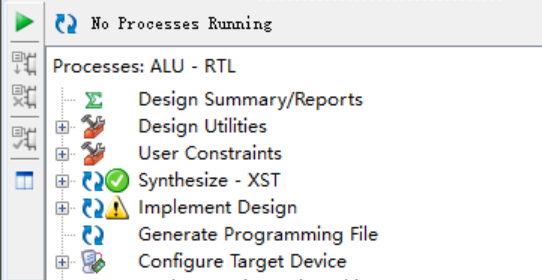
1）端口定义



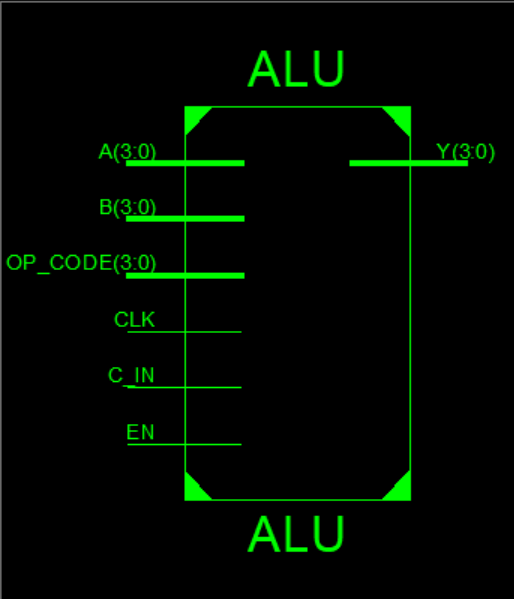
2）完成模块的结构体描述



3）语法检查



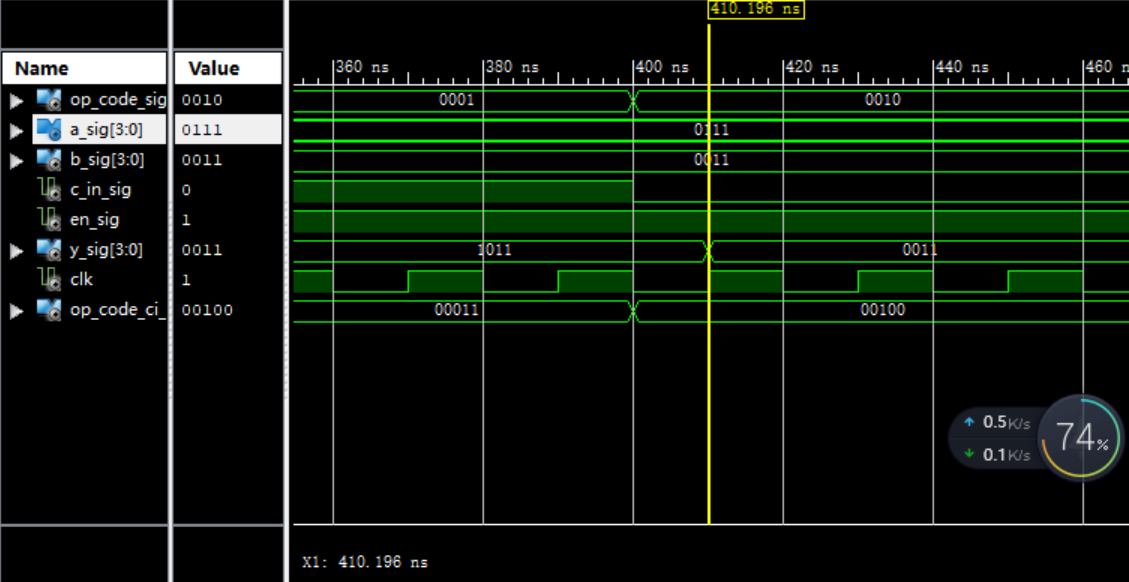
**RTL图：**

****

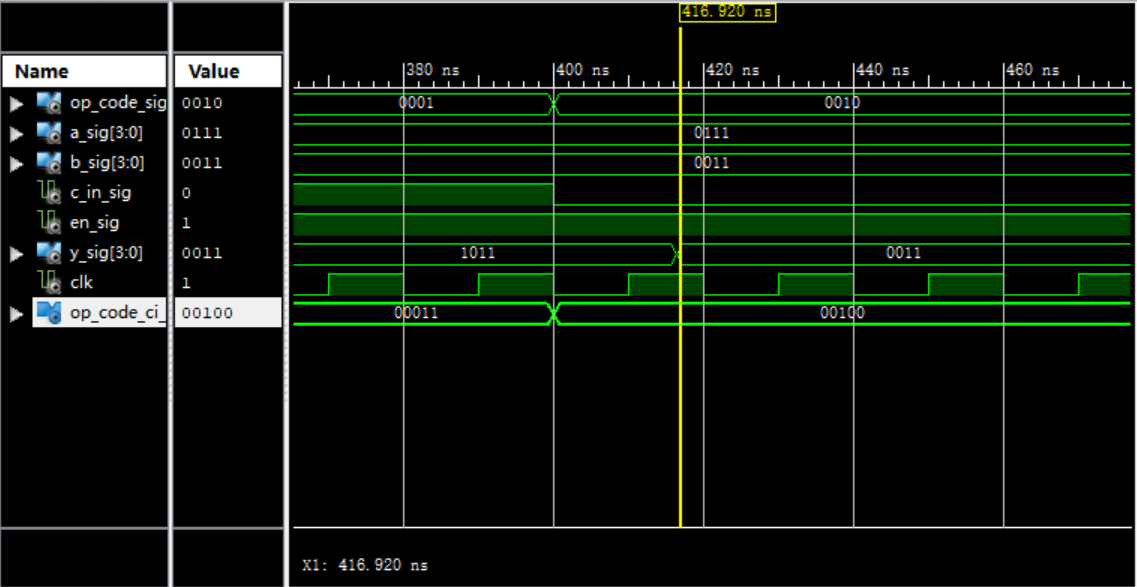
**4.创建测试平台文件**

**a.将模式设置为behavioral打开仿真工具**

**前仿真波形如下：**

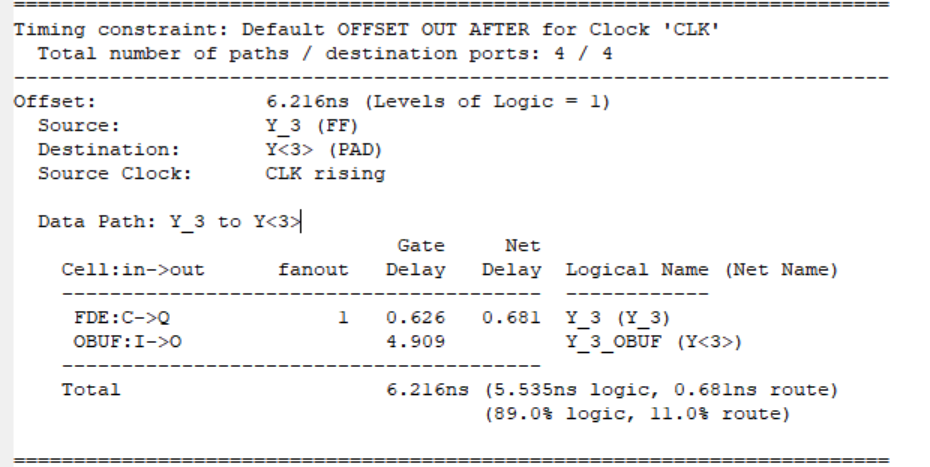


**后仿真波形如下：**



二者对比，可以明显地发现后仿真波形的信号具有一定的时延。

**实验资源分析报告：**



在综合报告中可以看到各类的时延，门时延以及布线时延。从静态分析报告中我们可以得到工作频率的最大值，即4.909ns。

**四、实验总结**

在本实验中，我们用case语句创建了计数器工程中的ALU模块，并为验证后续实验中的设计模板做好了准备。

在后续的模块中，将从MEM模块中读出用于计算的数据和期望的结果，将用于计算的输入数据传入ALU进行计算，计算的结果传到CMP的输入接口。在CMP中，ALU计算的结果将与MEM模块读出的结果进行比较，最后给出比较结果。

**五、实验代码**

**ALU.VHD：**

--导入库

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity ALU is

   port ( CLK           : in std\_logic ; --时钟信号

           OP\_CODE  : in std\_logic\_vector ( 3 downto 0 ) ;  --操作码

           A,B          : in std\_logic\_vector ( 3 downto 0 ) ;  --输入

            C\_IN        : in std\_logic ;                    --输入

             EN        : in std\_logic ;         --使能信号

            Y          : out std\_logic\_vector (3 downto 0) );        --输出

end entity ALU;

--ALU的RTL描述

architecture RTL of ALU is

    signal OP\_CODE\_CI : std\_logic\_vector ( 4 downto 0 ); --定义操作码信号

begin

    OP\_CODE\_CI <= OP\_CODE & C\_IN ;  --赋值

    process ( CLK ) begin

      if rising\_edge ( CLK ) then  --上升沿

    if (EN = '1') then

    --定义各种操作的具体步骤

   case OP\_CODE\_CI is

       when "00000" => Y <= A;

       when "00001" => Y <= A + 1 ;

       when "00010" => Y <= A + B ;

       when "00011" => Y <= A + B + 1;

       when "00100" => Y <= A + not B ;

       when "00101" => Y <= A + not B + 1;

       when "00110" => Y <= A - 1 ;

       when "00111" => Y <= A ;

        when "01000" => Y <= A and B ;

        when "01010" => Y <= A or B ;

        when "01100" => Y <= A xor B ;

        when "01110" => Y <= not A ;

        when "10000" => Y <= (others => '0') ;

        when others  => Y <= (others => 'X' );

    end case;

    end if;

  end if ;

end process;

end architecture RTL;

**ALU\_TB:**

--导入库

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

--定义实体

ENTITY ALU\_TB\_vhd IS

END ALU\_TB\_vhd;

ARCHITECTURE test OF ALU\_TB\_vhd IS

    COMPONENT alu  --组件映射

    PORT(   CLK : in std\_logic ;

          OP\_CODE : IN std\_logic\_vector(3 downto 0);

            A : IN std\_logic\_vector(3 downto 0);

            B : IN std\_logic\_vector(3 downto 0);

            C\_IN : IN std\_logic;

             EN : IN std\_logic;

            Y : OUT std\_logic\_vector(3 downto 0)

         );

    END COMPONENT;

    --定义对应的信号

    SIGNAL OP\_CODE\_SIG :  std\_logic\_vector(3 downto 0):= (others => '0');

    SIGNAL A\_SIG :  std\_logic\_vector(3 downto 0):= "0111" ;

    SIGNAL B\_SIG :  std\_logic\_vector(3 downto 0):= "0011" ;

    SIGNAL C\_IN\_SIG :  std\_logic := '0';

    SIGNAL EN\_SIG : std\_logic := '1';

    SIGNAL Y\_SIG :  std\_logic\_vector(3 downto 0);

    SIGNAL CLK :  std\_logic := '0' ;

    SIGNAL OP\_CODE\_CI\_SIG : std\_logic\_vector ( 4 downto 0 ) := (others => '0') ;

begin

      OP\_CODE\_SIG <= OP\_CODE\_CI\_SIG(4 downto 1); --高三位赋给 OP\_CODE\_SIG

      C\_IN\_SIG <= OP\_CODE\_CI\_SIG(0);            --0位赋给C\_IN\_SIG

    --端口映射

    uut: alu PORT MAP(

        OP\_CODE => OP\_CODE\_SIG,

        A => A\_SIG,

        B => B\_SIG,

        C\_IN => C\_IN\_SIG,

        Y => Y\_SIG,

        EN => EN\_SIG,

        CLK => CLK  );

  --时钟信号赋值

  clk <= not clk after 10 ns;

    --输入

  process

  begin

  OP\_CODE\_CI\_SIG <= "00000";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00001";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00010";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00011";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00100";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00101";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00110";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "00111";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "01000";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "01010";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "01100";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "01110";   wait for 100 ns;

  OP\_CODE\_CI\_SIG <= "10000";   wait;

end process;

END architecture test;

**六、实验的问题**

本次实验完成了书中 Lab6 算术逻辑单元 ALU 的 RTL 描述，在实验过程中学习了 VHDL 语言中 case 语句的描述。这次的实验的前仿真由于之前有所接触，所以可以较快地获得解决。对于后仿真部分，不知道是不是我本身软件的问题，无法从behavioral直接转到post-route进行仿真，需要重新配置文件进行，后仿真的过程中我也发现了后仿真所花的时间明显比前仿真多，不过这也是正常，毕竟它还考虑了具体布线时时延的部分。对于最高工作频率一开始不知道是什么东西，在老师的帮助下，才有了一点点粗浅的认识，也第一次学会看资源分析文件，但是对于资源分析文件的很多内容我都还很不理解，简单的说，就是看不太懂，希望随着学习的深入，能将其慢慢搞清楚。

**流程图：**

**开始**

**结束**

输出Y

Y

根据OP\_CODE操作

得到Y

N

Y

Y

N

**结束**

Clk上升？

**输入**

EN=1？