四位加法器实验报告

项晨东 2019011831

实验目的

1. 掌握组合逻辑电路的基本分析和设计方法。
2. 理解半加器和全加器的工作原理并掌握利用全加器构成不同字长的加法器的各种方 法。
3. 学习元件例化的方式进行硬件电路设计。
4. 学会利用软件仿真实现对数字逻辑电路的逻辑功能进行验证和分析。

实验内容

1. 设计实现逐次进位加法器, 进行软件仿真并在实验平台上进行测试。
2. 设计实现超前进位加法器, 进行软件仿真并在实验平台上进行测试。

实验代码以及说明

一位全加器

详细的电路原理在注释中

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

entity add1 is
  port(
    a,b,cin:in std_logic; --定义输入，分别为加数a，加数b，低位的进位
    s,cout:out std_logic; --定义输出，分别为结果s，对高位的进位
    p,g:buffer std_logic  --定义buffer，用于超前进位
  );
end add1;

architecture plus of add1 is
begin
  process(a,b)
  begin
    p<= a xor b; --进位传递信号，当加数中有1个为1时产生
    g<= a and b; --进位产生，当加数都为1时产生
  end process;

  process(cin, p,g)
  begin
    s<=p xor cin; --当前位的输出是传递和低位进位的异或
    cout<= g or (cin and p); --高位进位是由进位产生或者由进位传递产生
```

```
  end process;
end plus;
```

逐次进位四位加法器

详细的电路原理在注释中

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

entity add4 is
  port(
    a,b:in std_logic_vector(3 downto 0); --定义输入的四个位
    cin:in std_logic; --定义低位进位
    s:out std_logic_vector(3 downto 0); --定义输出四位
    cout:out std_logic --定义高位进位输出
    );
end add4;


architecture simple of add4 is
  component add1  --引用一位全加器
    port( a,b,cin:in std_logic;
        s,cout:out std_logic;
        p,g:buffer std_logic
    );
  end component;
  signal p,g,c:std_logic_vector(3 downto 0); --定义中间传递的信号
begin  --按顺序进行接口映射
  fa0: add1 port map(a(0),b(0),cin,s(0),c(0),p(0),g(0)); --输入输出一一对应
  fa1: add1 port map(a(1),b(1),c(0),s(1),c(1),p(1),g(1)); --低位进位来自于更低的位
  fa2: add1 port map(a(2),b(2),c(1),s(2),c(2),p(2),g(2)); --高位输出走向更高的位
  fa3: add1 port map(a(3),b(3),c(2),s(3),cout,p(3),g(3));
end simple;
```

超前进位四位加法器

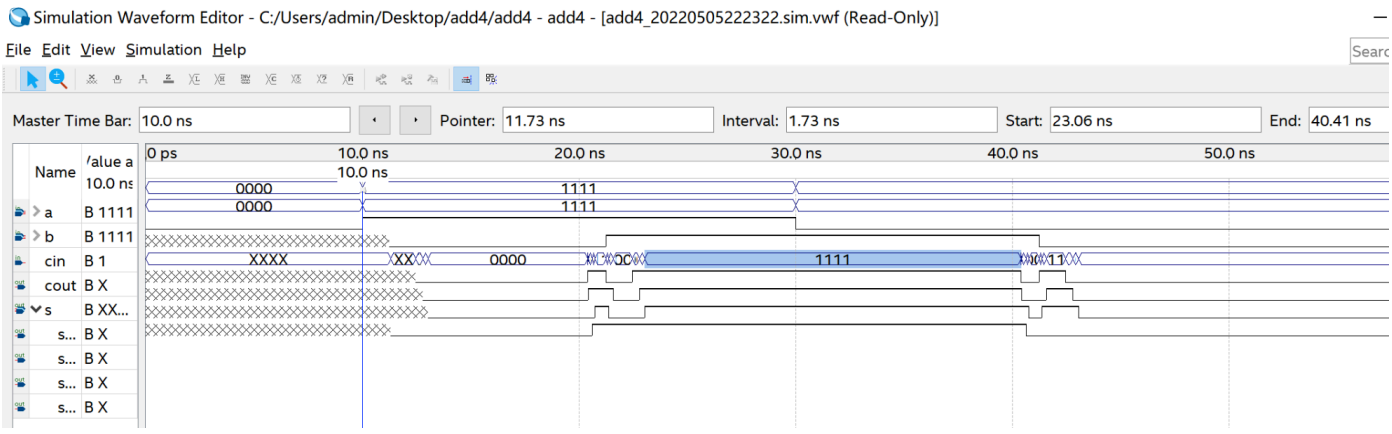
详细的电路原理在注释中

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

entity add4ad is
```

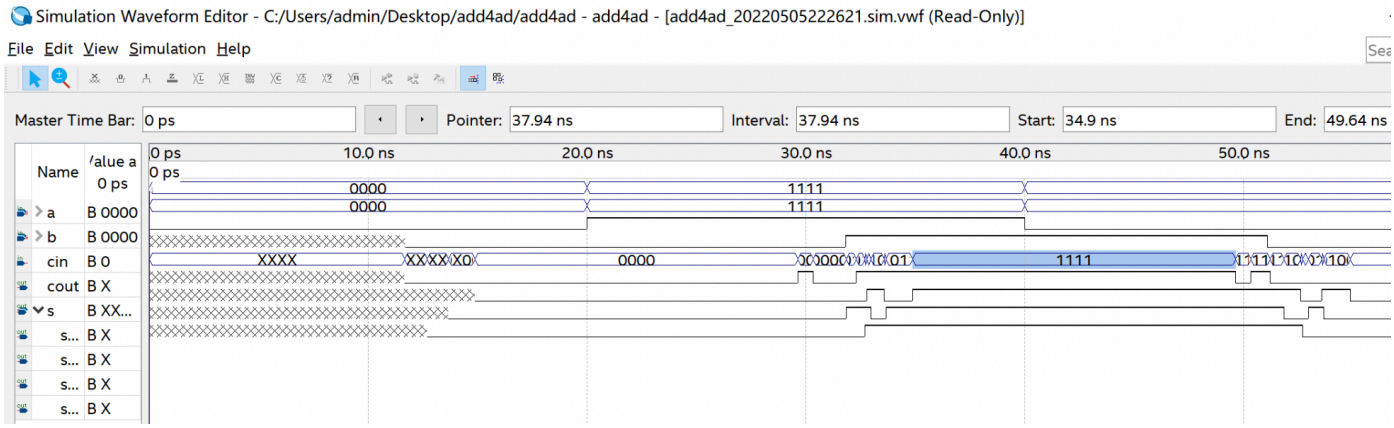
```
  port(a,b:in std_logic_vector(3 downto 0); --定义输入的四个位
    cin:in std_logic; --定义低位进位
    s:out std_logic_vector(3 downto 0); --定义输出四位
    cout:out std_logic --定义高位进位输出
    );
end add4ad;

architecture advance of add4ad is
  component add1 --引用一位全加器
    port( a,b,cin:in std_logic;
        s,cout:out std_logic;
        p,g:buffer std_logic
    );
  end component;
  signal p,g,c:std_logic_vector(3 downto 0);
begin
  fa0:add1 port map(a(0),b(0),cin, s=>s(0),p=>p(0),g=>g(0)); --由于是超前进位
  fa1:add1 port map(a(1),b(1),c(0),s=>s(1),p=>p(1),g=>g(1)); --进位由后续产生
  fa2:add1 port map(a(2),b(2),c(1),s=>s(2),p=>p(2),g=>g(2)); --所以跳过进位输出的映射
  fa3:add1 port map(a(3),b(3),c(2),s=>s(3),p=>p(3),g=>g(3));
  process(p,g)
  begin
    c(0)<=g(0) or (p(0) and cin); --依据超前进位的逻辑表达式
    c(1)<=g(1) or (p(1) and g(0)) or (p(1) and p(0) and cin); --依次表达出各个进位的表达式
    c(2)<=g(2) or (p(2) and g(1)) or (p(2) and p(1) and g(0))
            or (p(2) and p(1) and p(0) and cin);
    cout<=g(3) or (p(3) and g(2)) or (p(3) and p(2) and g(1))
               or (p(3) and p(2) and p(1) and g(0))
            or (p(3) and p(2) and p(1) and p(0) and cin);
  end process;
end advance;
```

仿真结果：

逐次进位加法器



超前进位加法器

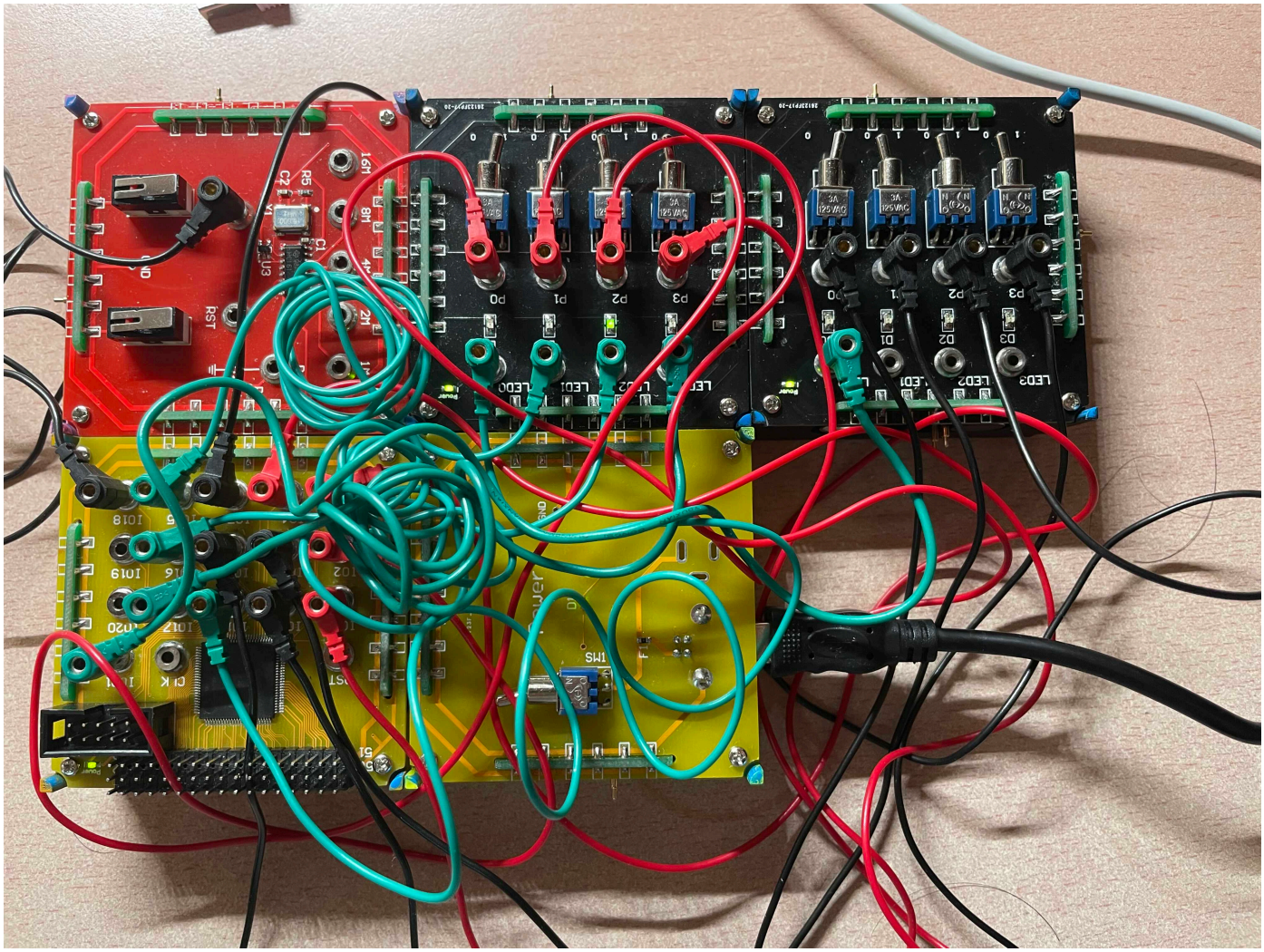


仿真结果的比较

按照输入变化从`0000+0000+0=0000`变化到`1111+1111+1=1111`。按照所有输出彻底稳定时为节点计算延时。可知逐次进位加加法器的延时在`13ns`左右, 超前进位加法器的延时在`15ns`左右。由于位数较少, 超前进位没有体现出明显的优势。

下载实验结果

左边四个按钮是其中一个加数, 从左自右依次低位到高位; 右边四个按钮是另外一个加数, 从左自右依次低位到高位。LED灯从左至右是低位到高位, 第五位是向高位的进位。 当前显示的是`0011 + 0001 =  0100`的结果, 即`3+1=4`。



总结和心得

1. 每个vhd文件最多只能有一个entity, 为了方便下载两个不同的逻辑电路, 比较好的方法是建立不同的工程文件, 而不是在工程文件中修改。
2. vhd的编译比较严格, 有时候多一个分号都不可以
3. 在写代码时要理清子组件之间的逻辑关系, 防止弄错

最后非常感谢老师和助教的辛苦付出。