# 实验三 简易模型机中组合器件的实现

# 班级 计科1903 姓名 陈旭 学号 201914020128

一、实验目的

1. 了解简易模型机的内部结构和工作原理。
2. 分析模型机的功能，设计 ALU 和移位逻辑。
3. 分析模型机的工作原理，设计模型机控制信号产生逻辑。

二、实验内容

1. 用 VHDL 语言设计ALU(函数发生器)。
2. 用 VHDL 语言设计移位逻辑。
3. 用 VHDL 语言设计控制信号产生逻辑。
4. 理解简易模型机的结构和工作原理。

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ALU(函数发生器)，移位逻辑，控制信号产生逻辑。
   * 采用的软件工具是Quartus II。
2. 实验软件操作步骤
3. ALU
4. 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\experiment3\alu设置project name为alu）-【next】（设置文件名alu）-【next】（设置芯片类型为【cyclone-EP1CT144C8】）-【finish】

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

1. 写好源代码，保存文件（alu.vhd）。
2. 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。
3. 波形仿真及验证。新建一个vector waveform file。按照程序所述插入S, M, inx， iny， Cf, Zf, outbus七个节点（inx，iny为八位输入节点向量，S 为四位输出节点向量， outbus为八位输出节点向量,Cf,Zf为输出结点，M为输入节点）。(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。任意设置inx, iny, S, M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。
4. 时序仿真和功能仿真。
5. 查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。
6. **移位逻辑**
7. 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\experiment3\rslogic设置project name为rslogic）-【next】（设置文件名rslogic）-【next】（设置芯片类型为【cyclone-EP1CT144C8】）-【finish】

(2).新建：【file】-【new】（VHDL File）-【OK】

1. 写好源代码，保存文件（rslogic.vhd）。
2. 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。
3. 波形仿真及验证。新建一个vector waveform file。按照程序所述插入fbus, frrbus, frlbus, inbus, outbus2, Cf六个节点（inbus为八位输入节点向量,outbus2为八位输出节点向量,Cf为输出结点，fbus, frrbus, frlbus为输入节点）。(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。任意设置inx, iny, S, M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。
4. 时序仿真和功能仿真。
5. 查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。
6. **控制信号产生逻辑**
7. 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\experiment3\signal\_generate设置project name为signal\_generate）-【next】（设置文件名signal\_generate）-【next】（设置芯片类型为【cyclone-EP1CT144C8】）-【finish】

(2).新建：【file】-【new】（VHDL File）-【OK】

1. 写好源代码，保存文件（rslogic.vhd）。
2. 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。
3. 波形仿真及验证。新建一个vector waveform file。按照程序所述插入 sm, mov1, mov2, mov3, add, sub, or1, not1, rsr, rsl, jmp, jz, jc, in1, out1, Zf, Cf, nop, halt，in\_en, out\_en, ldpc, inc, we, xl, dl, m, fbus, frr, frl, ld, Cf\_en, Zf\_en, Sm\_en，ir, wa, ra, madd, S等节点。(操作为：右击 -【insert】-【insert node or bus】-【node finder】（pins=all；【list】）-【>>】-【ok】-【ok】)。任意设置结点的输入波形…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。
4. 时序仿真和功能仿真。
5. 查看RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

四、实验过程

1. 编译过程

* **源代码（VHDL设计）如下**

1. ALU

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity alu is

    port(

        S: in std\_logic\_vector(3 downto 0);

        M: in std\_logic;

        inx, iny: in std\_logic\_vector(7 downto 0);

        Cf, Zf: out std\_logic:='0';

        outbus: out std\_logic\_vector(7 downto 0):="00000000");

end alu;

architecture alu of alu is

    signal result:std\_logic\_vector(8 downto 0):="000000000";

    signal sa, sb:std\_logic\_vector(8 downto 0);

    begin

        sa<='0'&inx;

        sb<='0'&iny;

        process(S, M, inx, iny)

        begin

            if (M='0') then --math compute or zhitong

                if (S="1001") then

                    outbus<=inx+iny;

                    result<=sa+sb;

                    Cf<=result(8);

                    if (result="000000000") then

                        Zf<='1';

                    else Zf<='0';

                    end if;

                elsif(S="0110") then

                    outbus<=iny-inx;

                    result<=sb-sa;

                    Cf<=result(8);

                    if(result="000000000") then

                        Zf<='1';

                    else Zf<='0';

                    end if;

                elsif(S="1010") then

                    Cf<='0';

                    Zf<='0';

                    outbus<=iny;

                elsif(S="0100") then

                    Cf<='0';

                    Zf<='0';

                    outbus<=iny;

                else

                    outbus<=iny;

                    Cf<='0';

                    Zf<='0';

                end if;

            else --logic compute

                if(S="1011") then

                    outbus<= inx or iny;

                    Cf<='0';

                    Zf<='0';

                elsif(S="0101") then

                    outbus<=not iny;

                    Cf<='0';

                    Zf<='0';

                end if;

            end if;

        end process;

    end architecture alu;

1. **移位逻辑**

library ieee;

use ieee.std\_logic\_1164.all;

entity rslogic is

    port(

        fbus, frlbus, frrbus:in std\_logic;

        inbus:in std\_logic\_vector(7 downto 0);

        outbus2: out std\_logic\_vector(7 downto 0);

        Cf:out std\_logic:='0'

    );

end rslogic;

architecture rslogic of rslogic is

    begin

        process(fbus, frlbus, frrbus)

            begin

            if(fbus='1') then

                outbus2<=inbus;

                Cf<='0';

            else

                if (frlbus='1') then

                    outbus2<=inbus(6 downto 0)&inbus(7);

                    Cf<=inbus(7);

                elsif(frrbus='1') then

                    outbus2<=inbus(0)&inbus(7 downto 1);

                    Cf<=inbus(0);

                else

                    outbus2<="ZZZZZZZZ";

                    Cf<='0';

                end if;

            end if;

            end process;

    end architecture rslogic;

1. **控制信号产生逻辑**

library ieee;

use ieee.std\_logic\_1164.all;

entity signal\_generate is

    port(

        sm, mov1, mov2, mov3, add, sub, or1, not1, rsr, rsl, jmp, jz, jc, in1, out1, Zf, Cf, nop, halt: in std\_logic;

        ir:in std\_logic\_vector(7 downto 0);

        ra, wa, madd:out std\_logic\_vector(1 downto 0);

        s: out std\_logic\_vector(3 downto 0);

        in\_en, out\_en, ldpc, inc, we, xl, dl, m, fbus, frr, frl, ld, Cf\_en, Zf\_en, Sm\_en: out std\_logic

    );

end signal\_generate;

architecture signal\_generate of signal\_generate is

    signal in0: std\_logic:='1';

    begin

        process(sm, mov1, mov2, mov3, add, or1, sub, not1, rsr, rsl, jmp, jz, jc, in1, out1, Zf, Cf, nop, halt, ir)

        begin

            ld<=not sm;

            ldpc<=(jc and Cf) or (jz and Zf) or jmp;

            inc <= (jz and (not Zf)) or (jc and (not Cf)) or nop or (not sm);

            if (ir(7 downto 4)="0010") then

                in0<='1';

            else

                in0<='0';

            end if;

            we <= not(mov1 or mov3 or add or sub or or1 or not1 or rsr or rsl or in0) or (not sm);

            ra <= ir(1 downto 0);

            wa <= ir(3 downto 2);

            if (sm='1' and mov3='1') then

                madd<="01";

            elsif (sm='1' and mov2='1') then

                madd<="10";

            else

                madd<="00";

            end if;

            s<=ir(7 downto 4);

            xl<=mov2;

            dl<=mov3 or jmp or (Zf and jz) or (Zf and jc) or (not sm);

            if ((ir(7 downto 4)="1001") or (ir(7 downto 4)="0110") or (ir(7 downto 4)="1010") or (ir(7 downto 4)="1111") or (ir(7 downto 4)="0100")) then

                m<='0';

            else

                m<='1';

            end if;

            frl<=rsl;

            frr<=rsr;

            fbus<= mov1 or mov2 or add or sub or or1 or not1 or rsr or rsl;

            Cf\_en<=add or sub or or1 or rsr or rsl;

            Zf\_en<=add or sub or or1 ;

            Sm\_en<=not halt;

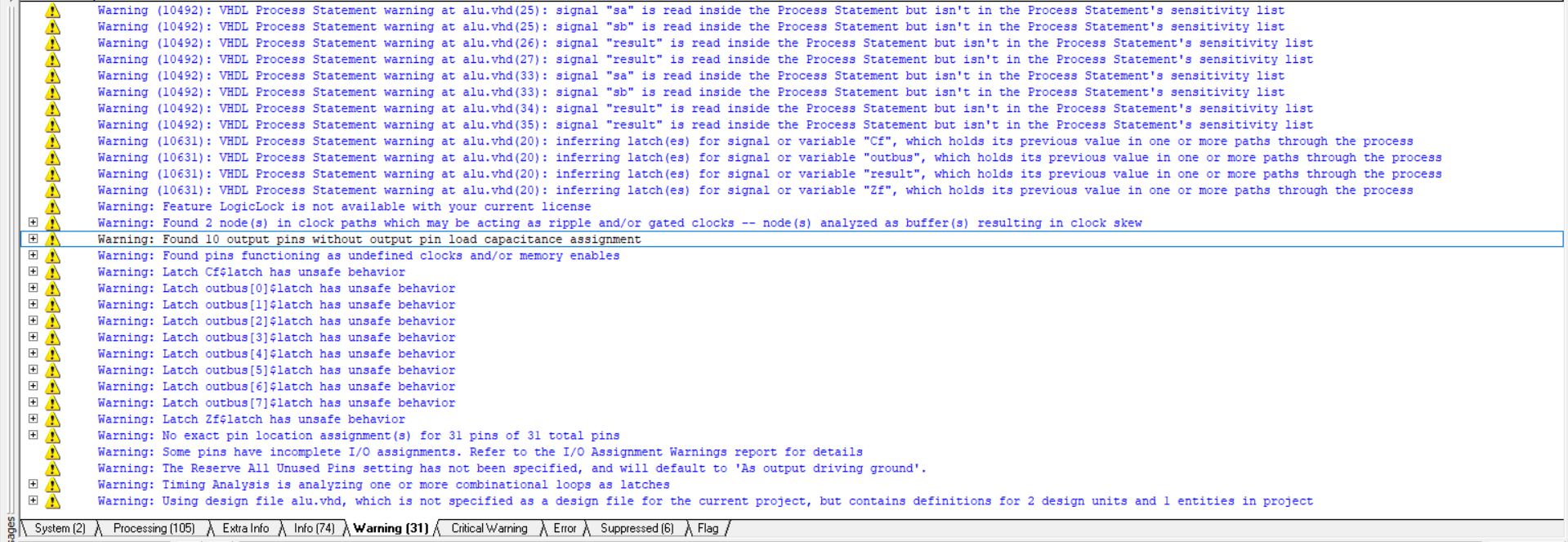
            in\_en<=in1;

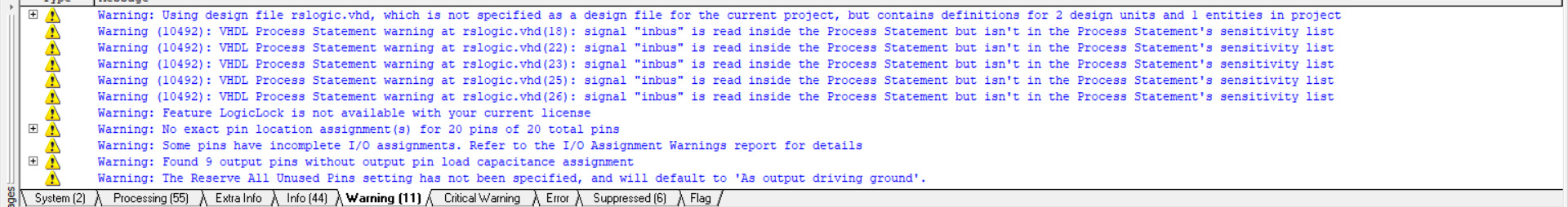
            out\_en<=out1;

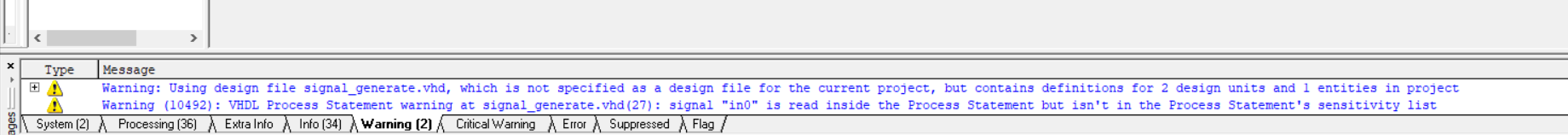
        end process;

    end architecture signal\_generate;

**b)编译、调试过程**







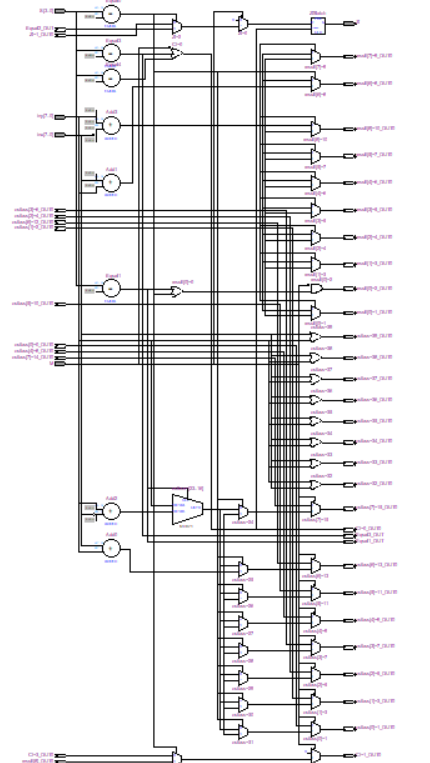
自上至下依次为ALU, 移位逻辑, 控制信号产生逻辑的编译器提示信息。

三者编译器均给出警告，但均能够编译通过。

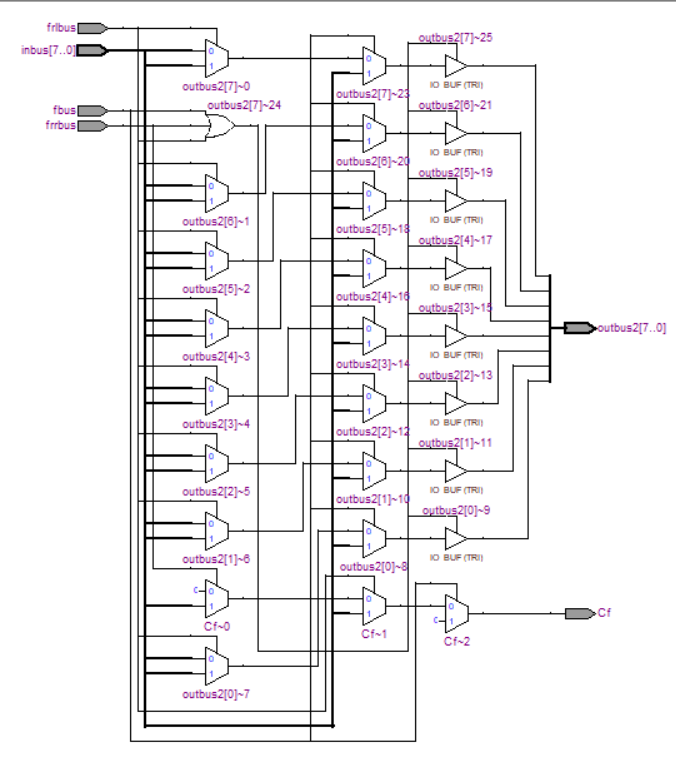
三者资源消耗基本都趋近于0.

**c) RTL视图**

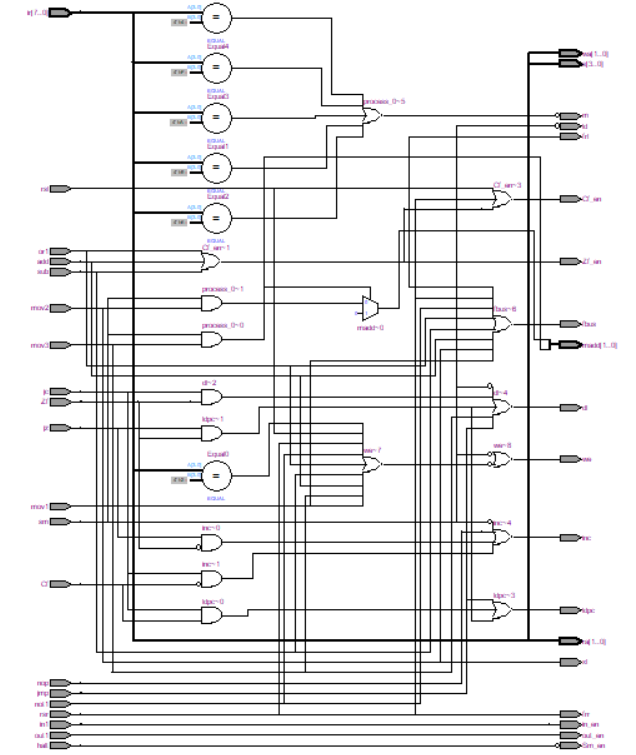
1. ALU



1. 位移逻辑



1. 控制信号产生逻辑



**d)结果分析及结论**

* ALU

当输入M为0，执行算术运算或者直通；此时

当S为“1001”执行inx+iny;

当S为“0110”执行iny-inx;

当S为“1001”、“1010”、“0100”或者其他，直通。

当输入M为1，执行逻辑运算；此时

当S为“1011”执行inx or iny;

当S为“0110”执行not iny;

* 移位逻辑

当输入fbus为1时，直通。

当输入fbus为0时：

frlbus为1时循环左移

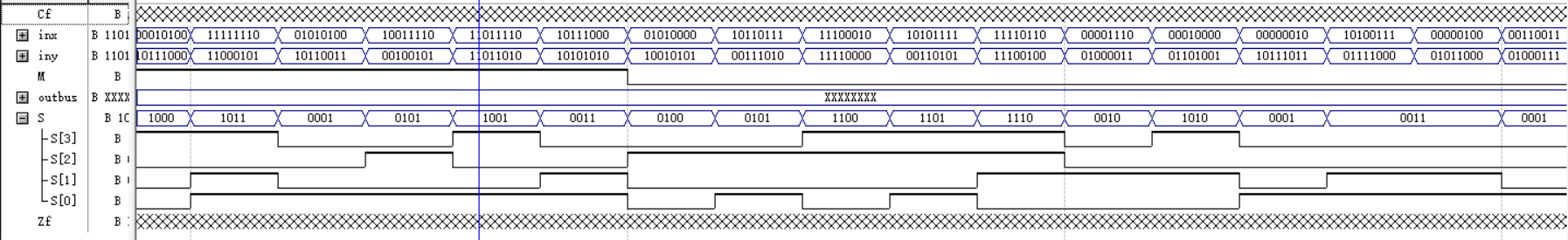
frrbus为0时循环右移

否则输出为高阻态(“ZZZZZZZZ”)

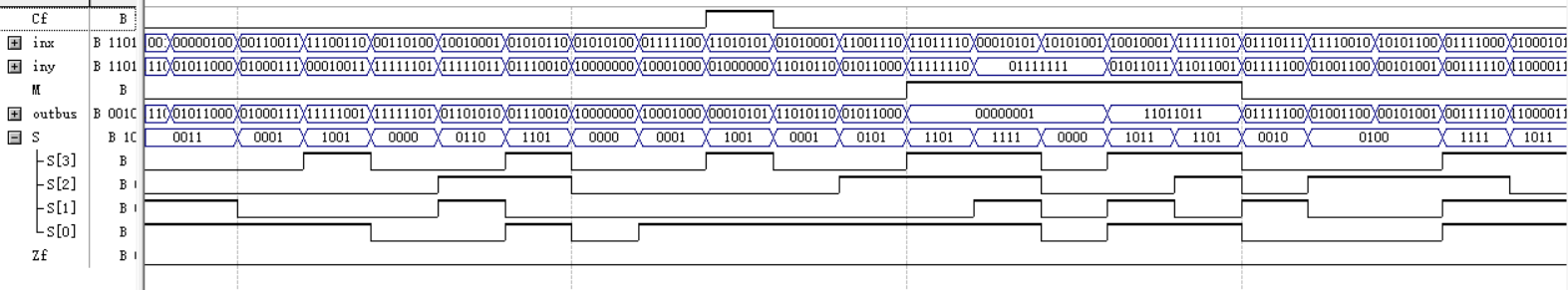
* 控制信号产生逻辑

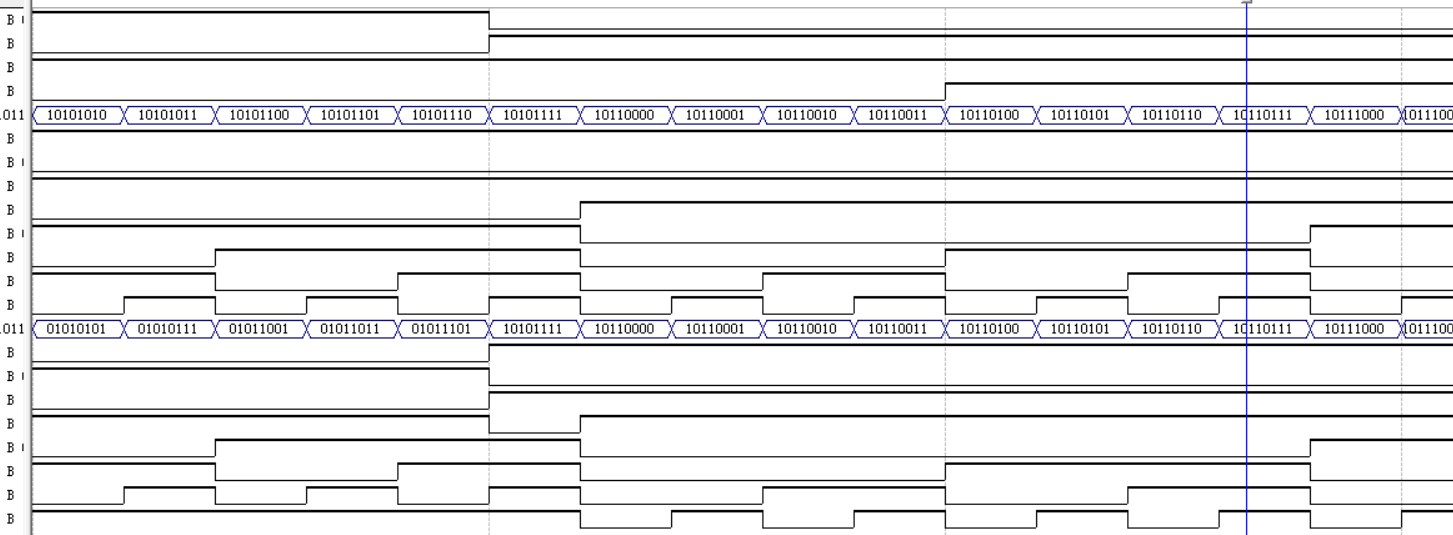
根据输入信号产生按表格对应的控制信号，如ld为not sm;等

1. 波形仿真
2. **ALU**
3. 波形仿真过程（过程详见实验步骤）



1. 波形仿真波形图





1. 结果分析及结论

以第一个周期为例

当输入M为0，执行算术运算或者直通；此时

当S为“1001”执行inx+iny,输出正确

当S为“0110”执行iny-inx,输出正确

当S为“1001”、“1010”、“0100”或者其他，直通, 输出正确

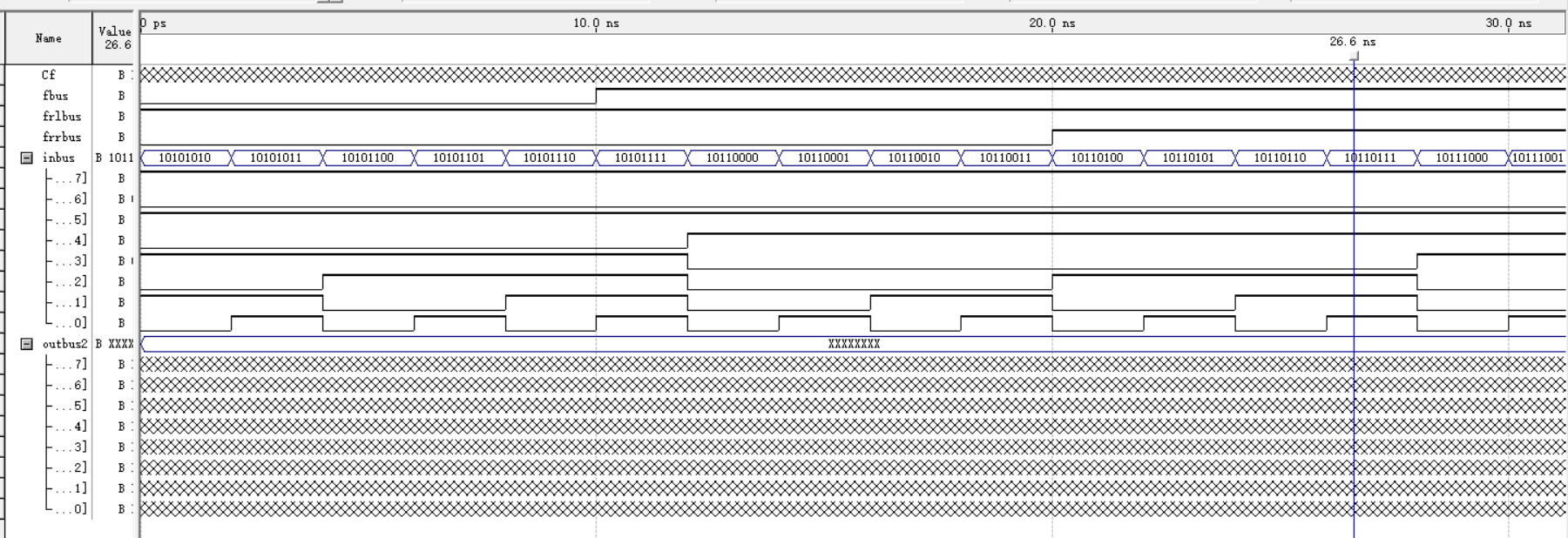
当输入M为1，执行逻辑运算；此时

当S为“1011”执行inx or iny, 输出正确;

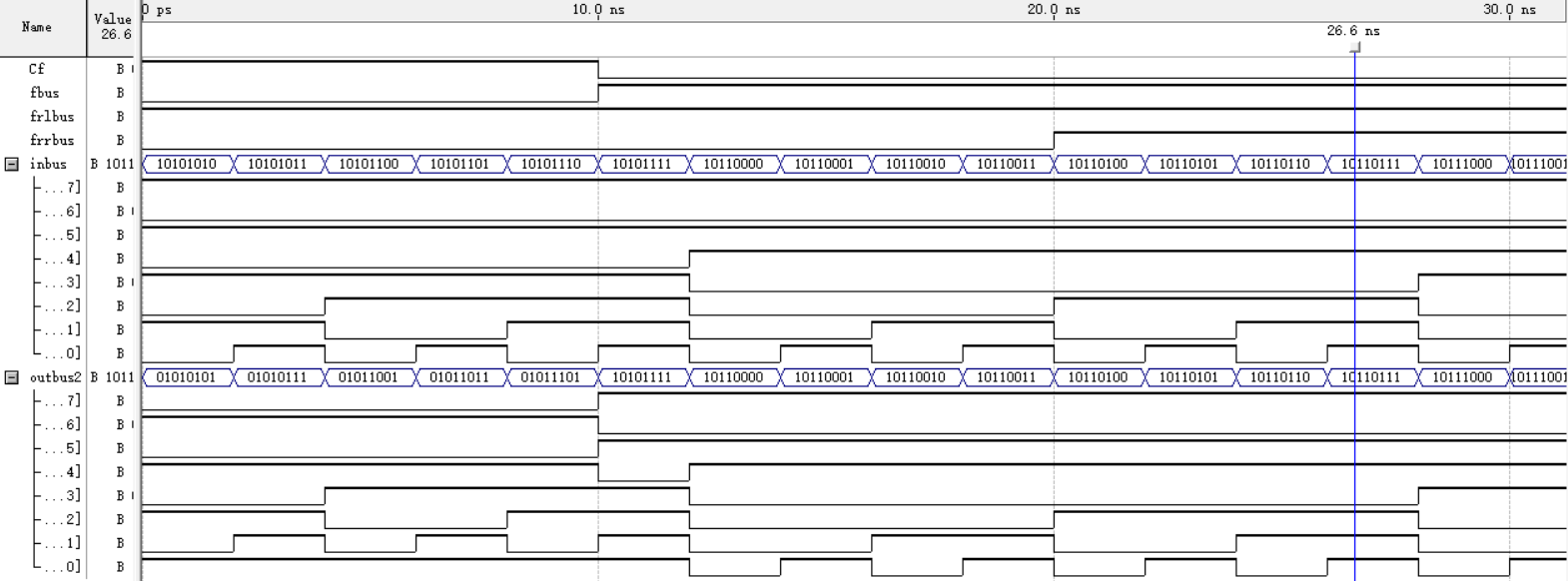
当S为“0110”执行not iny;, 输出正确

结果正确

1. **移位逻辑**
2. 波形仿真过程（过程详见实验步骤）



1. 波形仿真波形图



1. 结果分析及结论

由上图得：

当输入fbus为1时，直通，输出正确

当输入fbus为0时：

frlbus为1时循环左移，输出正确

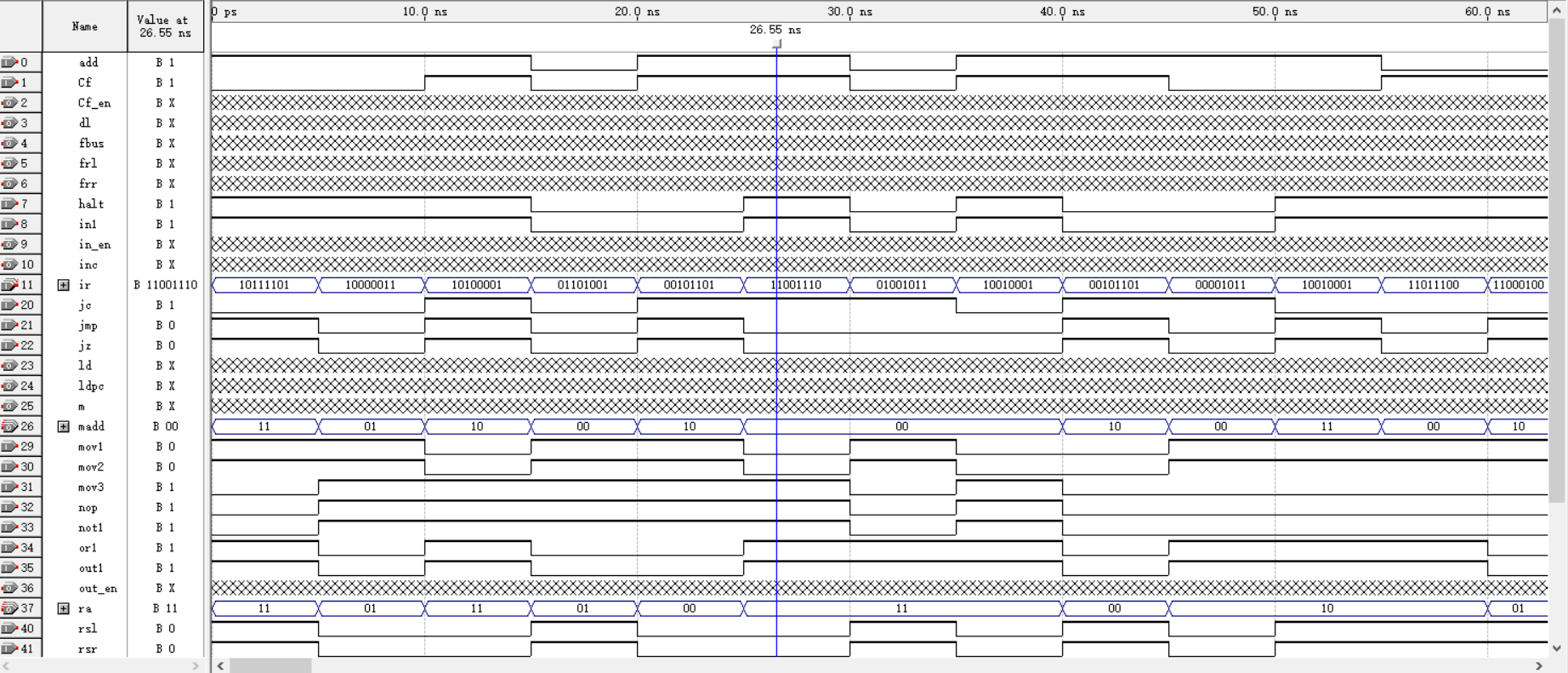
frrbus为0时循环右移，输出正确

否则输出为高阻态(“ZZZZZZZZ”)，输出正确(上图因为图片大小等原因未能显示，

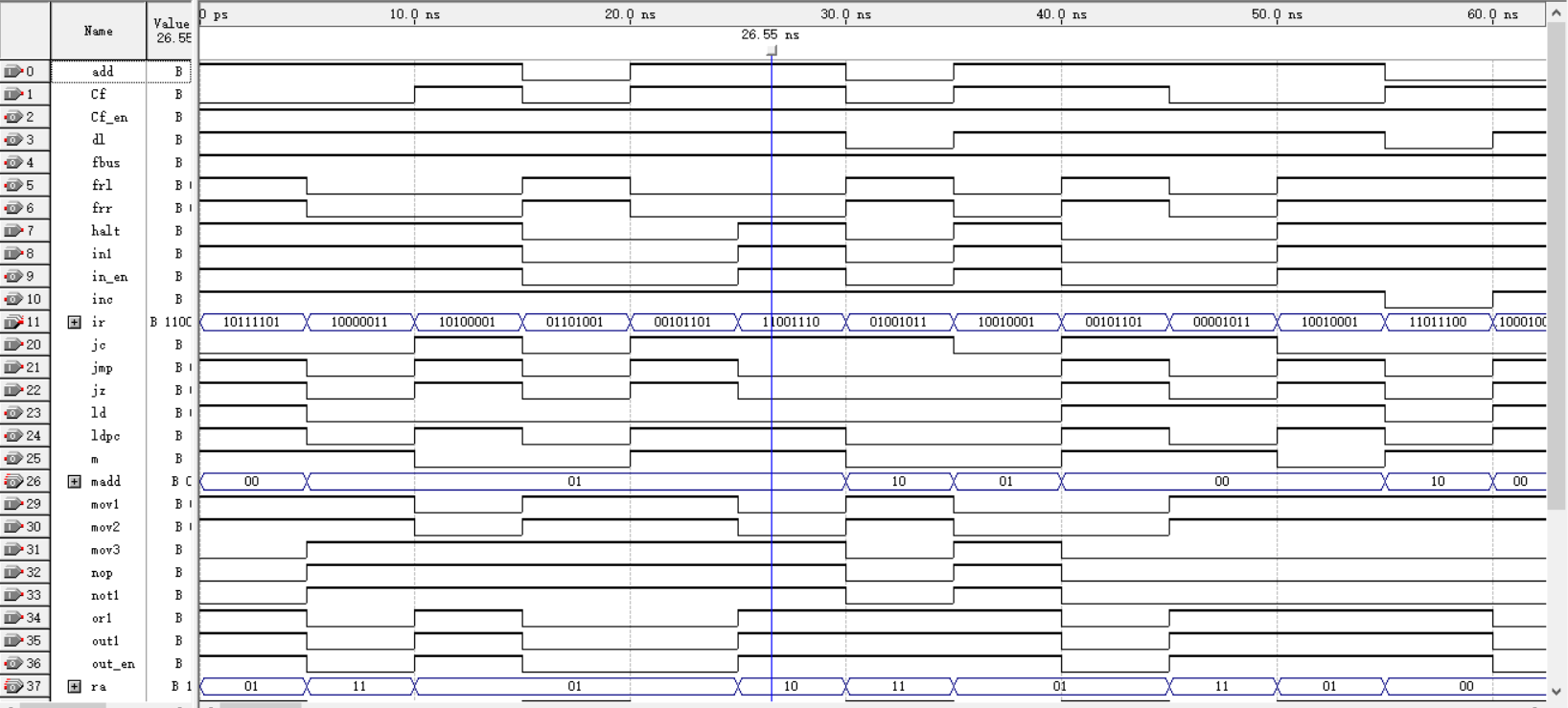
但是输出的黑色粗线的确存在)

结果正确

1. **控制信号产生逻辑**
2. 波形仿真过程（过程详见实验步骤）



1. 波形仿真波形图





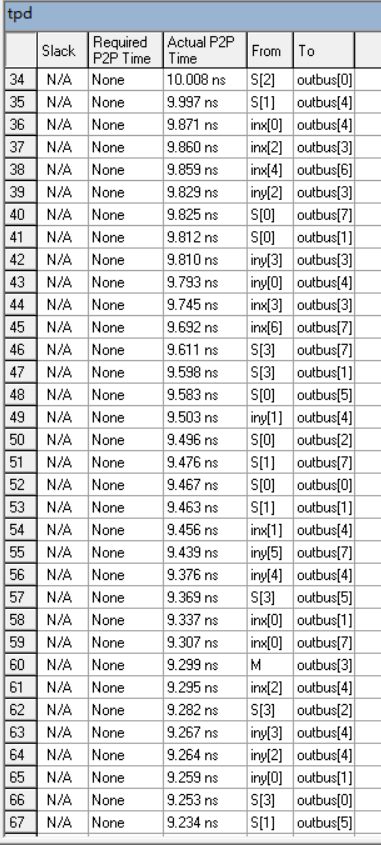
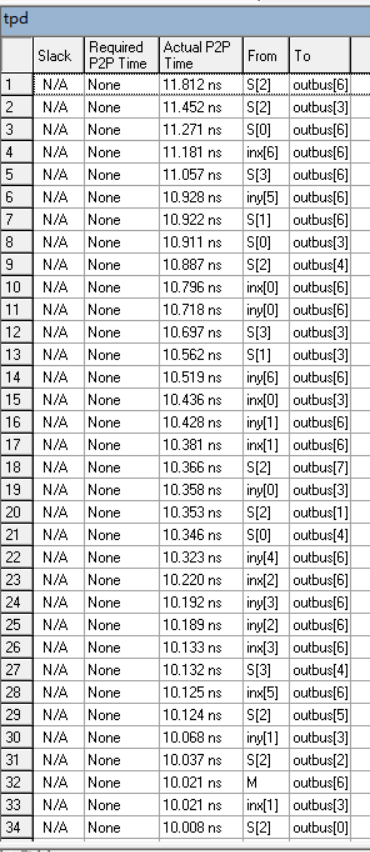
1. 时序仿真

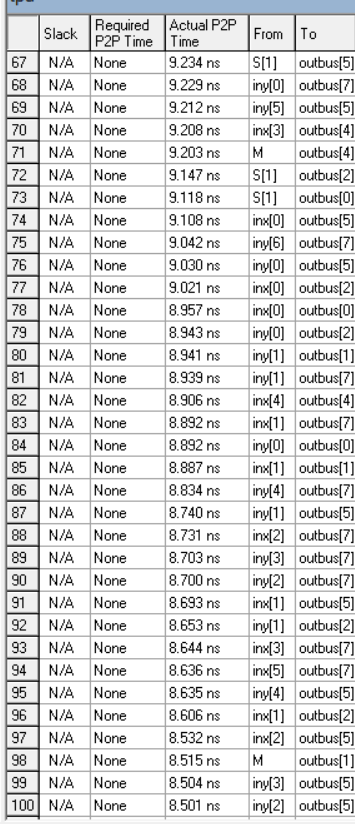
* ALU

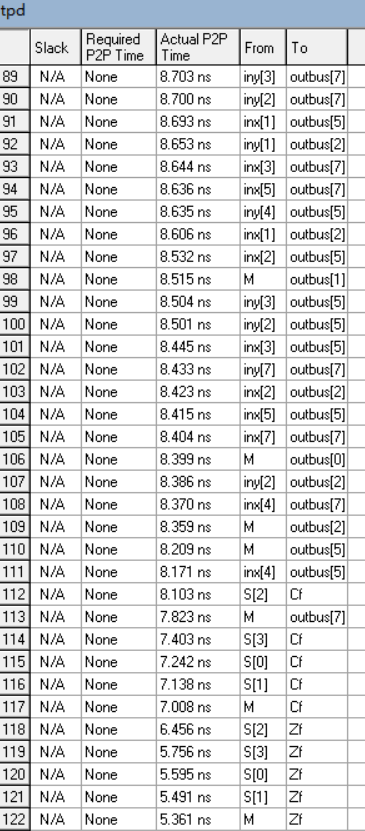
1. 时序仿真过程

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）

1. 时序仿真图







1. 结果分析及结论

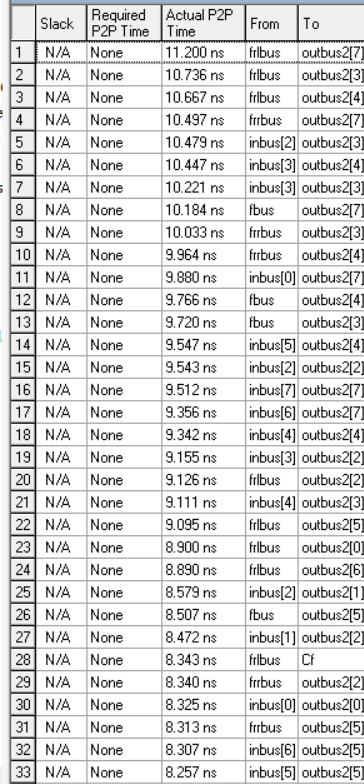
每个引脚只间相互传递产生的延时各不相同，挑选其中p2p时间的最大值，为S [2]传递给outbus[6]。为11.812ns，故整体延时为11.812ns。

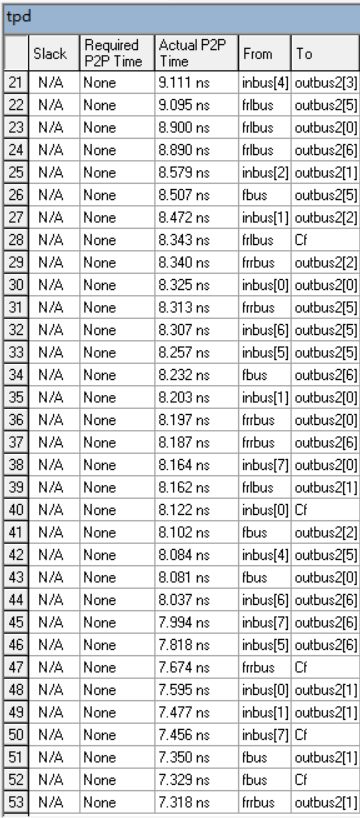
* 移位逻辑

1. 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）

1. 时序仿真图





1. 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frlbus传递给outbus2[7]。为11.200ns，故整体延时为11.200ns。

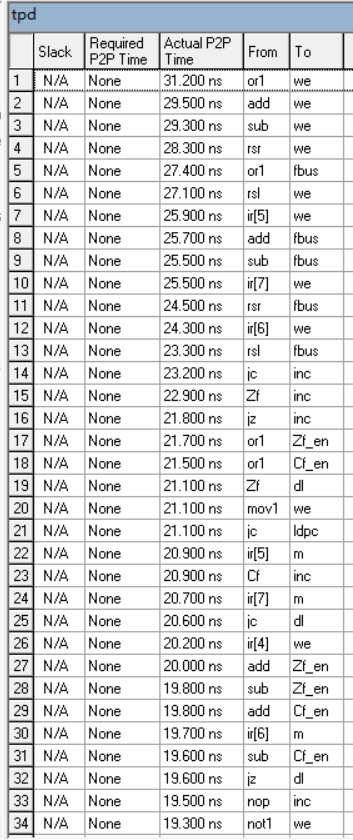
tpd (引脚到引脚的延时)

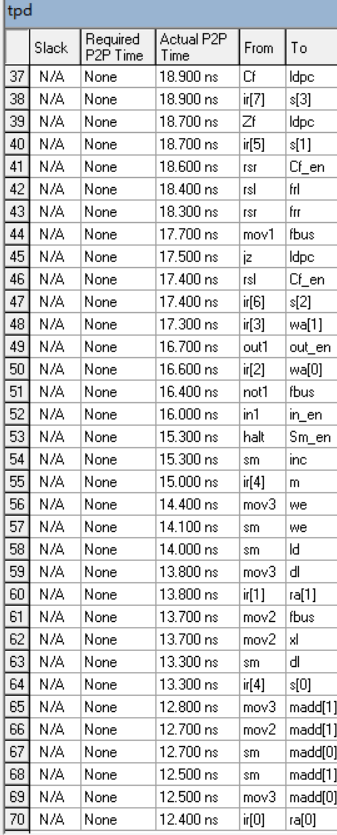
* 控制信号产生逻辑

1. 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）

1. 时序仿真图





1. 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or1传递给we。为31.200ns，故整体延时为31.200ns。

tpd (引脚到引脚的延时)

五、实验结论

1、思考题

1. 输出为高阻态至数据总线。按照上图分析，当不工作时各传入信号均为0，此时输出为高阻态，此时移位逻辑所在的线路无法通过，数据在通用寄存器和RAM等之间移动。
2. 使用触发器将其存储起来，便于后续处理和控制其他模块
3. 可以使用基础逻辑门来将某信号产生条件根据其逻辑函数表达式连接起来，然后根据输出波形随输入的变化而变化的情况来判断其正确性即可
4. 实验总结与实验心得

本次实验的内容与要求是简易模型机中组合器件的VHDL语言实现以及理解简易模型机的结构和工作原理。涉及的能力有VHDL组合逻辑电路编程能力，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，同时为时序电路的设计与编程实现奠定了基础。我因此受益良多。