# 实验三 加法器、运算器的实现

班级 计科 1808 姓名 张继伟 学号 201808010829

## 一、实验目的

1．了解运算器的内部结构。
2．熟悉并行加法器和串行加法器的工作原理。
3．分析模型机的功能，设计 ALU 和移位逻辑。

## 二、实验内容

1．用 VHDL 语言设计 4 位并行加法器
2．用 VHDL 语言设计 4 位串行加法器
3．用 VHDL 语言设计模型机运算器 ALU
4．用 VHDL 语言设计移位逻辑

## 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。
采用的软件工具是 Quartus II。

### 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】-【new project wizard】-【next】（设置文件路径+设置 project name 为 xor2）-【next】（设置文件名 xor2.vhd—在【add】)-【properties】(type=AHDL）-【next】(family=FLEX10K；name=EPF10K10TI144-4）-【next】-【finish】

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

2、写好源代码，保存文件（xor2.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，编译成功。

4、波形仿真及验证。新建一个 vector waveform file。按照程序所述插入 a,b,c 三个节点（a、b 为输入节点，c 为输出节点）。(操作为：右击 -【insert】-【insert node or bus】-【node finder】(pins=all；【list】）-【>>】-【ok】-【ok】)。任意设置 a,b 的输入波形…点击保存按钮保存。(操作为：点击 name（如：A））-右击-【value】-【clock】(如设置 period=200；offset=0)，同理设置 name B（如 120，,60)，保存)。然后【start simulation】，出 name C 的输出图。

5、时序仿真或功能仿真。

6、查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

## 四、实验过程

# 4 位并行加法器

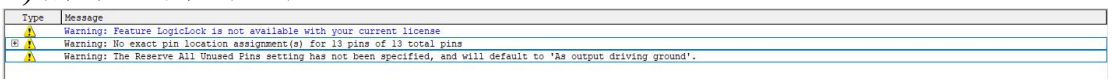
### 1、编译过程

a）源代码如图（VHDL 设计）

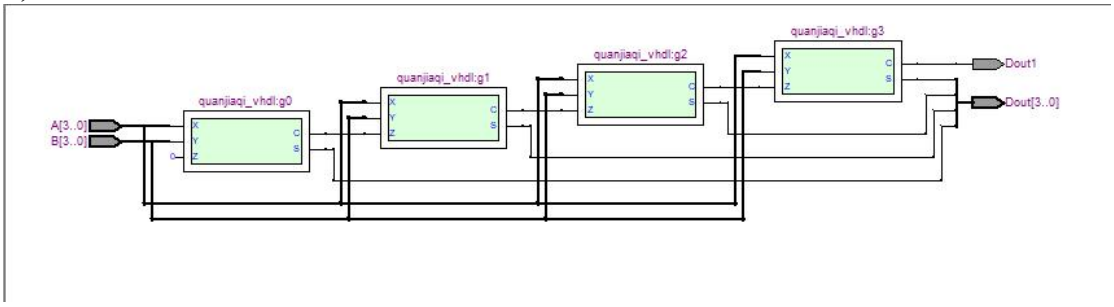
```
library ieee;
use ieee.std_logic_1164.all;
entity zjw_bingxingjiafaqi is
    port (A,B: in std_logic_vector(3 downto 0);
    Dout: out std_logic_vector(3 downto 0);
    Dout1:out std_logic);
end zjw_bingxingjiafaqi;

architecture bx of zjw_bingxingjiafaqi is
component quanjiaqi_vhdl
    port (X,Y,Z: in std_logic;
    S,C: out std_logic);
end component;
signal D: std_logic_vector(2 downto 0);
begin
    g0: quanjiaqi_vhdl port map(A(0),B(0),'0',Dout(0),D(0));
    g1: quanjiaqi_vhdl port map(A(1),B(1),D(0),Dout(1),D(1));
    g2: quanjiaqi_vhdl port map(A(2),B(2),D(1),Dout(2),D(2));
    g3: quanjiaqi_vhdl port map(A(3),B(3),D(2),Dout(3),Dout1);
end bx;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity quanjiaqi_vhdl is
    port (X,Y,Z: in std_logic;
    S,C: out std_logic);
end quanjiaqi_vhdl;

architecture quanjia of quanjiaqi_vhdl is
begin
S <= (X xor Y) xor Z;
C <= (X and Y) or (Z and (X xor Y));
end quanjia;
```

b)编译、调试过程




c) RTL 视图



d)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。

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

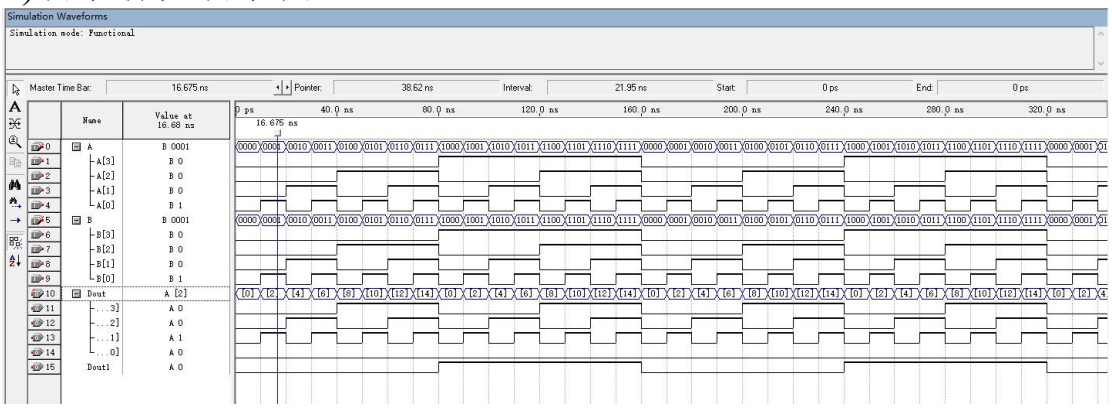
出一位结果，最后一个全加器额外输出一个进位。

例如输入“0000”和“1111”，输出结果为‘0’和“1111”;

**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

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

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

**3、 时序仿真**

a) 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）

b)时序仿真图

Registered Performance | tpd | tsu | tco | th | Custom Delays

| | Slack | Required P2P Time | Actual P2P Time | From | To |
|---|---|---|---|---|---|
| 1 | N/A | None | 11.975 ns | A[1] | Dout1 |
| 2 | N/A | None | 11.534 ns | B[0] | Dout1 |
| 3 | N/A | None | 11.138 ns | A[0] | Dout1 |
| 4 | N/A | None | 10.905 ns | B[2] | Dout1 |
| 5 | N/A | None | 10.890 ns | A[2] | Dout1 |
| 6 | N/A | None | 10.722 ns | B[0] | Dout[0] |
| 7 | N/A | None | 10.572 ns | A[1] | Dout[2] |
| 8 | N/A | None | 10.529 ns | B[1] | Dout1 |
| 9 | N/A | None | 10.331 ns | A[0] | Dout[0] |
| 10 | N/A | None | 10.131 ns | B[0] | Dout[2] |
| 11 | N/A | None | 10.063 ns | A[1] | Dout[3] |
| 12 | N/A | None | 9.955 ns | B[3] | Dout1 |
| 13 | N/A | None | 9.888 ns | A[3] | Dout1 |
| 14 | N/A | None | 9.735 ns | A[0] | Dout[2] |
| 15 | N/A | None | 9.622 ns | B[0] | Dout[3] |
| 16 | N/A | None | 9.509 ns | B[2] | Dout[2] |
| 17 | N/A | None | 9.489 ns | A[2] | Dout[2] |
| 18 | N/A | None | 9.377 ns | A[1] | Dout[1] |
| 19 | N/A | None | 9.226 ns | A[0] | Dout[3] |
| 20 | N/A | None | 9.126 ns | B[1] | Dout[2] |
| 21 | N/A | None | 8.993 ns | B[2] | Dout[3] |
| 22 | N/A | None | 8.978 ns | A[2] | Dout[3] |
| 23 | N/A | None | 8.941 ns | B[0] | Dout[1] |
| 24 | N/A | None | 8.617 ns | B[1] | Dout[3] |
| 25 | N/A | None | 8.550 ns | A[0] | Dout[1] |
| 26 | N/A | None | 8.044 ns | B[3] | Dout[3] |
| 27 | N/A | None | 7.974 ns | A[3] | Dout[3] |
| 28 | N/A | None | 7.932 ns | B[1] | Dout[1] |

b) 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

# 4 位串行加法器

**1，编译过程**

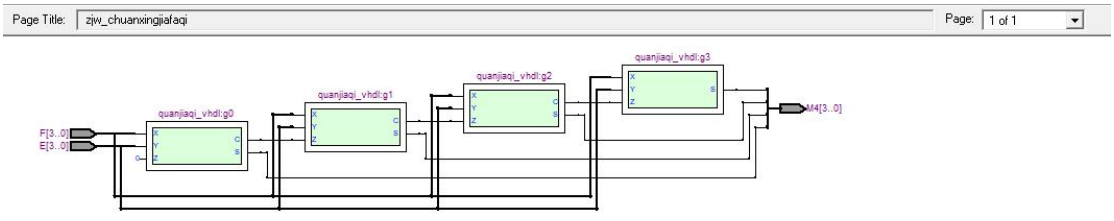
a）源代码如图（VHDL 设计）

```
library ieee;
use ieee.std_logic_1164.all;
entity zjw_chuanxingjiafaqi is
    port (F,E: in std_logic_vector(3 downto 0);
    M4: out std_logic_vector(3 downto 0));
end zjw_chuanxingjiafaqi;
architecture bre of zjw_chuanxingjiafaqi is
component quanjiaqi_vhdl
    port (X,Y,Z: in std_logic;
    S,C: out std_logic);
end component;
component yiwei_vhdl
    port (A: in std_logic_vector(3 downto 0);
          C: in std_logic;
          B: out std_logic_vector(3 downto 0));
end component;
signal  D: std_logic_vector(3 downto 0);
signal  N: std_logic_vector(3 downto 0);
signal  M0: std_logic_vector(3 downto 0);
signal  M1: std_logic_vector(3 downto 0);
signal  M2: std_logic_vector(3 downto 0);
signal  M3: std_logic_vector(3 downto 0);
signal  N0: std_logic_vector(3 downto 0);
signal  N1: std_logic_vector(3 downto 0);
signal  N2: std_logic_vector(3 downto 0);
signal  N3: std_logic_vector(3 downto 0);
signal  Dout: std_logic_vector(3 downto 0);
begin
    M0 <= F;
    N0 <= E;
g0: quanjiaqi_vhdl port map(M0(0),N0(0),'0',Dout(0),D(0));
    M1 <= Dout(0) & M0(3 downto 1);
    N1 <= Dout(0) & N0(3 downto 1);

g1: quanjiaqi_vhdl port map(M1(0),N1(0),D(0),Dout(1),D(1));
    M2 <= Dout(1) & M1(3 downto 1);
    N2 <= Dout(1) & N1(3 downto 1);

g2: quanjiaqi_vhdl port map(M2(0),N2(0),D(1),Dout(2),D(2));
    M3 <= Dout(2) & M2(3 downto 1);
    N3 <= Dout(2) & N2(3 downto 1);

g3: quanjiaqi_vhdl port map(M3(0),N3(0),D(2),Dout(3),D(3));
    M4 <= Dout(3) & M3(3 downto 1);

end ;
```

b)编译、调试过程

| Type | Message |
|---|---|
| ⚠ | Warning: Feature LogicLock is not available with your current license |
| ⚠ | Warning: No exact pin location assignment(s) for 12 pins of 12 total pins |
| ⚠ | Warning: The Reserve All Unused Pins setting has not been specified, and will default to 'As output driving ground'. |

c) RTL 视图



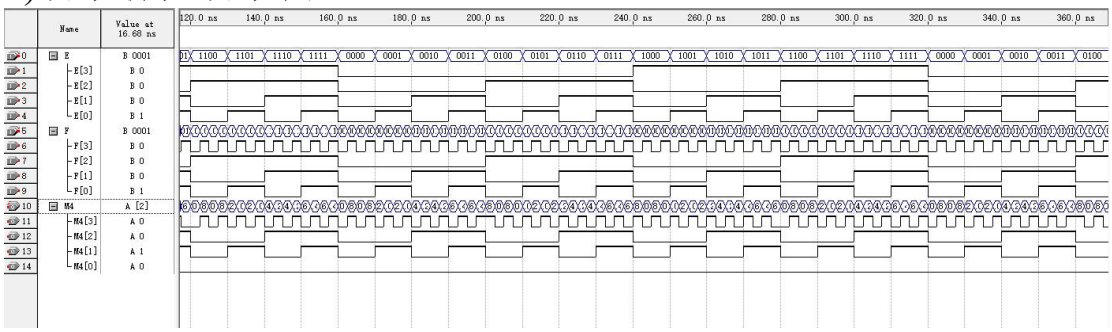
结果分析及结论

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 Q【3..0】和 Q【3..0】。

**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

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

**3,时序仿真**

d) 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）

b)时序仿真图

Registered Performance | tpd | tsu | tco | th | Custom Delays

| | Slack | Required P2P Time | Actual P2P Time | From | To |
|---|---|---|---|---|---|
| 1 | N/A | None | 10.737 ns | E[1] | M4[3] |
| 2 | N/A | None | 10.162 ns | F[1] | M4[3] |
| 3 | N/A | None | 9.977 ns | E[1] | M4[2] |
| 4 | N/A | None | 9.632 ns | E[1] | M4[1] |
| 5 | N/A | None | 9.520 ns | E[0] | M4[3] |
| 6 | N/A | None | 9.402 ns | F[1] | M4[2] |
| 7 | N/A | None | 9.242 ns | F[0] | M4[3] |
| 8 | N/A | None | 9.191 ns | F[2] | M4[3] |
| 9 | N/A | None | 9.056 ns | F[1] | M4[1] |
| 10 | N/A | None | 8.760 ns | E[0] | M4[2] |
| 11 | N/A | None | 8.663 ns | E[2] | M4[3] |
| 12 | N/A | None | 8.615 ns | F[3] | M4[3] |
| 13 | N/A | None | 8.532 ns | E[3] | M4[3] |
| 14 | N/A | None | 8.482 ns | F[0] | M4[2] |
| 15 | N/A | None | 8.430 ns | F[2] | M4[2] |
| 16 | N/A | None | 8.416 ns | E[0] | M4[1] |
| 17 | N/A | None | 8.212 ns | E[0] | M4[0] |
| 18 | N/A | None | 8.138 ns | F[0] | M4[1] |
| 19 | N/A | None | 7.934 ns | F[0] | M4[0] |
| 20 | N/A | None | 7.903 ns | E[2] | M4[2] |

e) 结果分析及结论

时序仿真不仅反应出输出和输入的逻辑关系，同时还计算了时间的延时信息，是与实际系统更接近的一种仿真结果。不过，要注意的是，这个时间延时是仿真软件“估算”出来的。

# 模型机运算器 ALU

**1,编译过程**

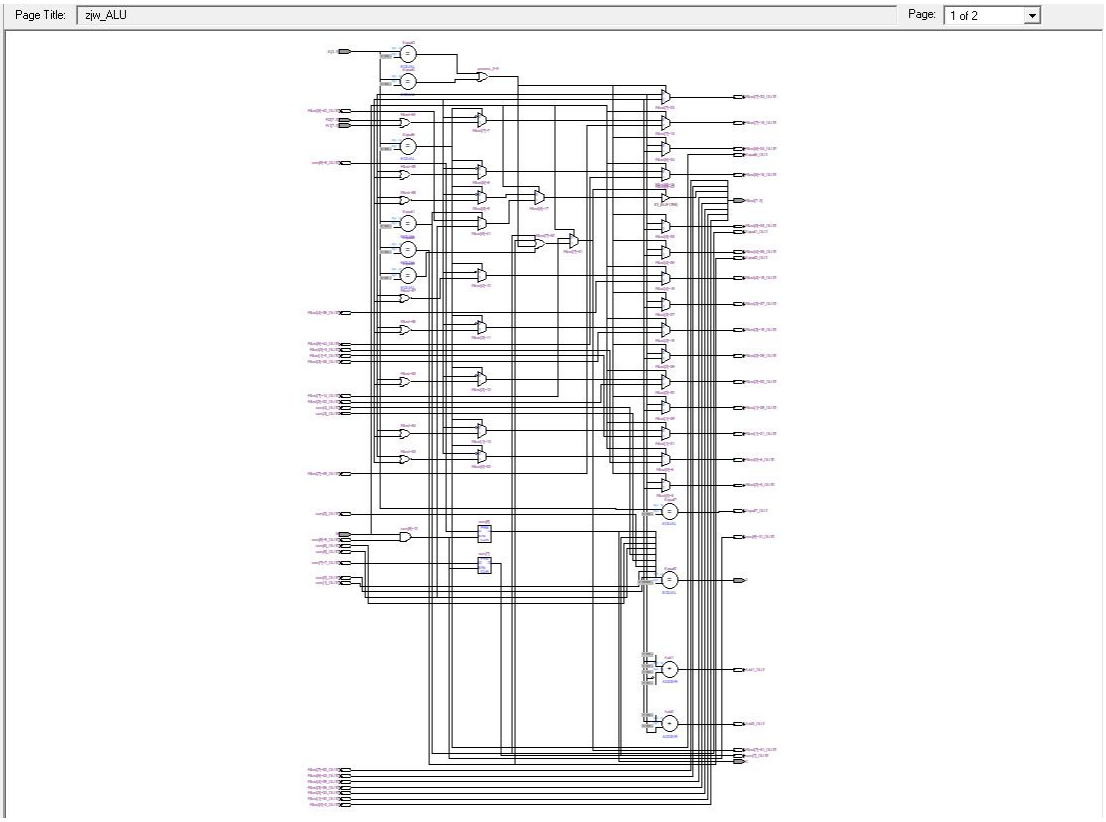
a）源代码如图（VHDL 设计）

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  entity zjw_ALU is
5  port(R1, R2: in std_logic_vector(7 downto 0);
6       S: in std_logic_vector(3 downto 0);
7       M:in std_logic;
8       C, Z: out std_logic;
9       Rlout: out std_logic_vector(7 downto 0));
10 end zjw_ALU;
11
12 architecture bev of zjw_ALU is
13 signal instruct: std_logic_vector(3 downto 0);
14 signal sum: std_logic_vector(8 downto 0);
15 signal d: std_logic;
16 begin
17 instruct <= S;
18 C <= sum(8);
19 Z <= '1' when sum="00000000" else '0';
20 process(instruct,M)
21 begin
22 if M='1' then
23     if instruct = "1001" then
24         sum <= ('0' & R1) + ('0' & R2);
25         Rlout <= sum(7 downto 0);
26     elsif instruct = "0110" then
27         sum <= ('0' & R1) - ('0' & R2);
28         Rlout <= sum(7 downto 0);
29     elsif instruct = "1010" or instruct = "0100" then
30         Rlout <= R1;
31     elsif instruct = "1111" then
32         Rlout <= R2;
33     else
34         Rlout <= "ZZZZZZZZ";
35     end if;
36 else
37     if instruct = "1011" then
38         Rlout <= R1 or R2;
39     elsif instruct = "0101" then
40         Rlout <= not R1;
41     else
42         Rlout <= "ZZZZZZZZ";
43     end if;
44 end if;
45 end process;
46 end;
```

b)编译、调试过程

| Type | Message |
|---|---|
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(24): signal "R1" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(24): signal "R2" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(25): signal "sum" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(27): signal "R1" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(27): signal "R2" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(28): signal "sum" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(30): signal "R1" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(32): signal "R2" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(38): signal "R1" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(38): signal "R2" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10492): VHDL Process Statement warning at zjw_ALU.vhd(40): signal "R1" is read inside the Process Statement but isn't in the Process Statement's sensitivity list |
| ⚠ | Warning (10631): VHDL Process Statement warning at zjw_ALU.vhd(20): inferring latch(es) for signal or variable "sum", which holds its previous value in one or more paths through the process |
| ⚠ | Warning: Feature LogicLock is not available with your current license |
| ⚠ | Warning: No exact pin location assignment(s) for 31 pins of 31 total pins |
| ⚠ | Warning: The Reserve All Unused Pins setting has not been specified, and will default to 'As output driving ground'. |
| ⚠ | Warning: Timing Analysis is analyzing one or more combinational loops as latches |
| ⚠ | Warning: Found pins functioning as undefined clocks and/or memory enables |
| ⚠ | Warning: Found 3 node(s) in clock paths which may be acting as ripple and/or gated clocks -- node(s) analyzed as buffer(s) resulting in clock skew |

c) RTL 视图




d)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；

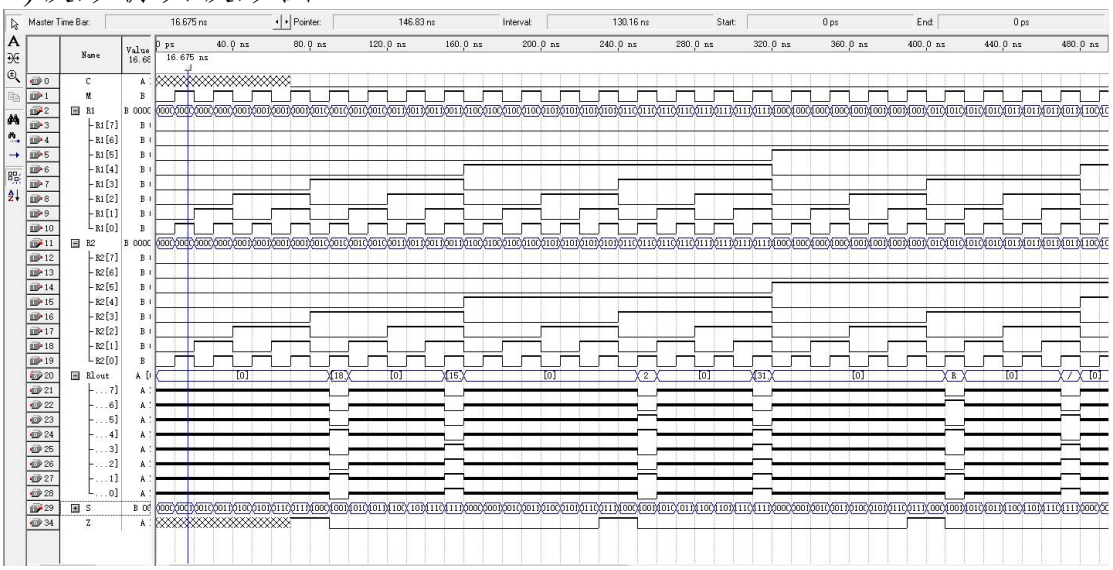
当前四位为“1010”，最后两位是“00”时，将后四位中的前两位做循环右移计算，保存到 R1 中，输出；

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；

**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

定制 8-1 多路复用器，本质上还是 8-1 多路复用器，只是得到方式是通过改变类属参量得到的，接收多个输入信号，按每个输入信号可恢复方式合成单个输出信号。

**3,时序仿真**

f) 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）

## b)时序仿真图

Registered Performance | tpd | tsu | tco | th | Custom Delays

| | Slack | Required P2P Time | Actual P2P Time | From | To |
|---|---|---|---|---|---|
| 1 | N/A | None | 24.054 ns | S[2] | Rlout[7] |
| 2 | N/A | None | 23.918 ns | S[2] | Rlout[3] |
| 3 | N/A | None | 23.897 ns | S[2] | Rlout[0] |
| 4 | N/A | None | 23.788 ns | S[0] | Rlout[7] |
| 5 | N/A | None | 23.652 ns | S[0] | Rlout[3] |
| 6 | N/A | None | 23.631 ns | S[0] | Rlout[0] |
| 7 | N/A | None | 23.629 ns | S[1] | Rlout[7] |
| 8 | N/A | None | 23.605 ns | S[2] | Rlout[2] |
| 9 | N/A | None | 23.493 ns | S[1] | Rlout[3] |
| 10 | N/A | None | 23.472 ns | S[1] | Rlout[0] |
| 11 | N/A | None | 23.339 ns | S[0] | Rlout[2] |
| 12 | N/A | None | 23.304 ns | S[2] | Rlout[4] |
| 13 | N/A | None | 23.300 ns | S[3] | Rlout[7] |
| 14 | N/A | None | 23.180 ns | S[1] | Rlout[2] |
| 15 | N/A | None | 23.164 ns | S[3] | Rlout[3] |
| 16 | N/A | None | 23.156 ns | S[2] | Rlout[1] |
| 17 | N/A | None | 23.143 ns | S[3] | Rlout[0] |
| 18 | N/A | None | 23.038 ns | S[0] | Rlout[4] |
| 19 | N/A | None | 23.028 ns | S[2] | Rlout[5] |
| 20 | N/A | None | 23.011 ns | S[2] | Rlout[6] |
| 21 | N/A | None | 22.890 ns | S[0] | Rlout[1] |
| 22 | N/A | None | 22.879 ns | S[1] | Rlout[4] |
| 23 | N/A | None | 22.851 ns | S[3] | Rlout[2] |
| 24 | N/A | None | 22.762 ns | S[0] | Rlout[5] |
| 25 | N/A | None | 22.745 ns | S[0] | Rlout[6] |
| 26 | N/A | None | 22.731 ns | S[1] | Rlout[1] |
| 27 | N/A | None | 22.603 ns | S[1] | Rlout[5] |
| 28 | N/A | None | 22.586 ns | S[1] | Rlout[6] |
| 29 | N/A | None | 22.550 ns | S[3] | Rlout[4] |
| 30 | N/A | None | 22.402 ns | S[3] | Rlout[1] |
| 31 | N/A | None | 22.274 ns | S[3] | Rlout[5] |
| 32 | N/A | None | 22.257 ns | S[3] | Rlout[6] |
| 33 | N/A | None | 20.723 ns | M | Rlout[7] |
| 34 | N/A | None | 20.574 ns | M | Rlout[0] |
| 35 | N/A | None | 20.381 ns | M | Rlout[3] |
| 36 | N/A | None | 20.068 ns | M | Rlout[2] |
| 37 | N/A | None | 19.783 ns | M | Rlout[4] |
| 38 | N/A | None | 19.619 ns | M | Rlout[1] |

## g) 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

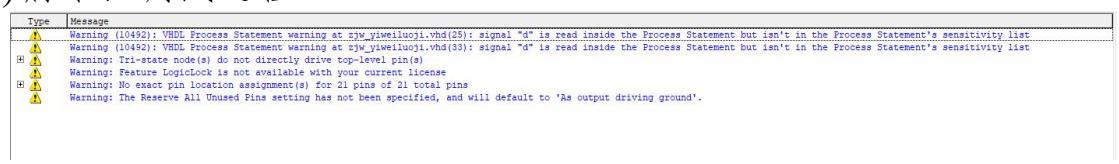
# 移位逻辑

**1,编译过程**

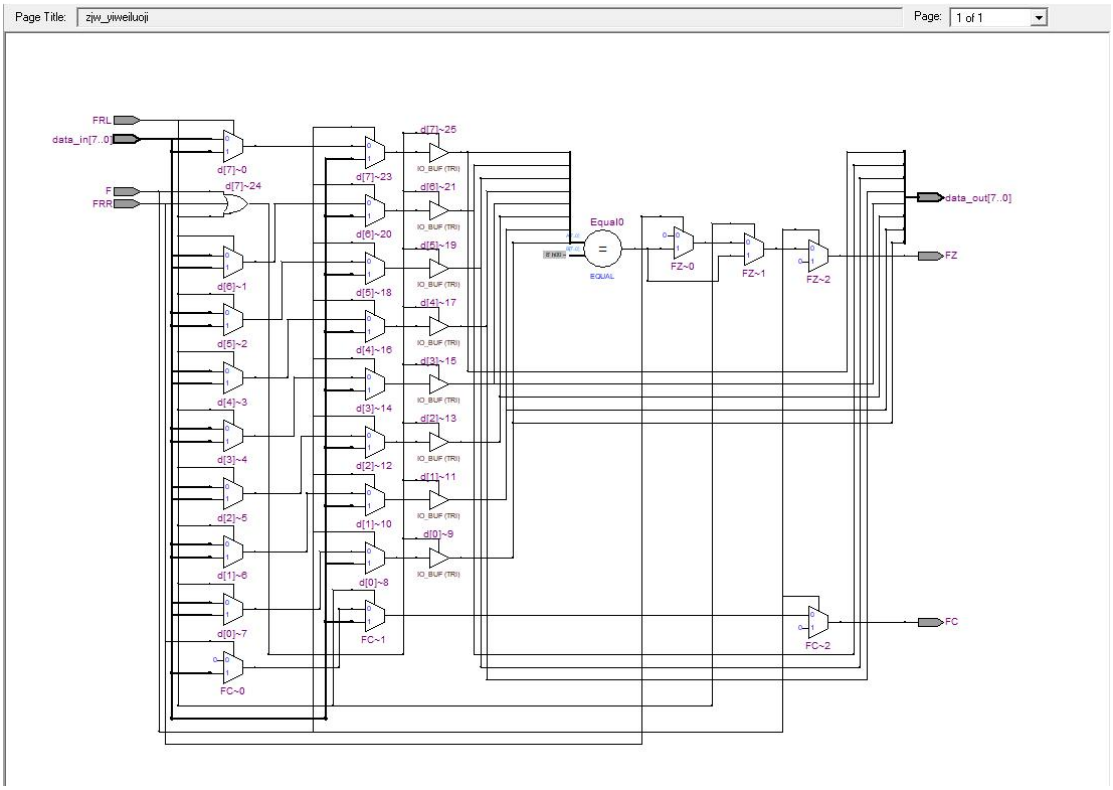
a）源代码如图（VHDL 设计）

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4
5   entity zjw_yiweiluoji is
6   port(F,FRL,FRR: in std_logic;
7        data_in: in std_logic_vector(7 downto 0);
8        FZ,FC: out std_logic;
9        data_out: out std_logic_vector(7 downto 0));
10  end zjw_yiweiluoji;
11
12  architecture behave of zjw_yiweiluoji is
13  signal d: std_logic_vector(7 downto 0);
14  begin
15       data_out<=d;
16       process(F,FRL,FRR,data_in)
17       begin
18       if F='1' then
19            d <=data_in;
20            FC <= '0';
21            FZ <= '0';
22       elsif FRL='1' then
23            d <=data_in(6 downto 0)&data_in(7);
24            FC <= data_in(7);
25            if d="00000000" then
26                 FZ <= '1';
27            else
28                 FZ <= '0';
29            end if;
30       elsif FRR='1' then
31            d <=data_in(0) & data_in(7 downto 1);
32            FC <= data_in(0);
33            if d="00000000" then
34                 FZ <= '1';
35            else
36                 FZ <= '0';
37            end if;
38       else
39            d <="ZZZZZZZZ";
40            FC <= '0';
41            FZ <= '0';
42       end if;
43       end process;
44  end;
```

b)编译、调试过程



c) RTL 视图



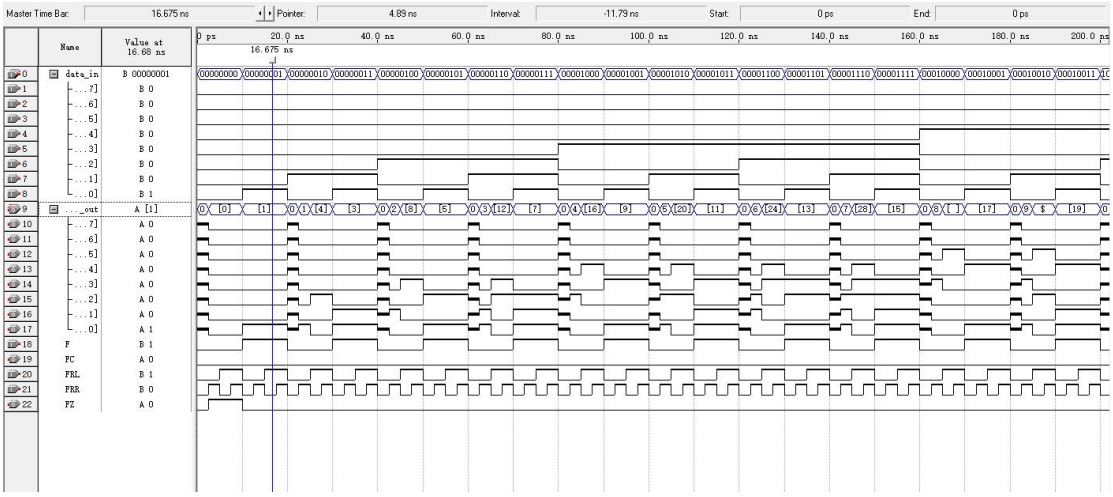

d)结果分析及结论

数据通路确定后,就可以设计指令系统中每条指令所需要的机器周期数。对于微程序控制的计算机,根据总线结构,需考虑哪些微操作可以安排在同一条微指令中,哪些微操作不能安排在同一条微指令中。

**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

**3,时序仿真**

h) 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）

b)时序仿真图

| | Slack | Required P2P Time | Actual P2P Time | From | To |
|---|---|---|---|---|---|
| 1 | N/A | None | 18.187 ns | data_in[4] | data_out[5] |
| 2 | N/A | None | 17.952 ns | data_in[5] | data_out[6] |
| 3 | N/A | None | 17.941 ns | data_in[4] | FZ |
| 4 | N/A | None | 17.528 ns | data_in[7] | FZ |
| 5 | N/A | None | 17.435 ns | data_in[5] | FZ |
| 6 | N/A | None | 17.355 ns | data_in[7] | data_out[6] |
| 7 | N/A | None | 16.974 ns | data_in[4] | data_out[3] |
| 8 | N/A | None | 16.729 ns | data_in[5] | data_out[5] |
| 9 | N/A | None | 16.360 ns | FRL | FZ |
| 10 | N/A | None | 16.108 ns | FRL | data_out[6] |
| 11 | N/A | None | 15.836 ns | FRL | data_out[5] |
| 12 | N/A | None | 15.820 ns | FRL | data_out[1] |
| 13 | N/A | None | 15.730 ns | data_in[7] | data_out[0] |
| 14 | N/A | None | 15.578 ns | data_in[5] | data_out[4] |
| 15 | N/A | None | 15.558 ns | FRL | data_out[2] |
| 16 | N/A | None | 15.447 ns | data_in[4] | data_out[4] |
| 17 | N/A | None | 14.756 ns | data_in[0] | FZ |
| 18 | N/A | None | 14.627 ns | data_in[3] | FZ |
| 19 | N/A | None | 14.625 ns | FRL | data_out[3] |
| 20 | N/A | None | 14.562 ns | FRL | data_out[0] |
| 21 | N/A | None | 14.516 ns | data_in[2] | FZ |
| 22 | N/A | None | 14.479 ns | FRR | data_out[6] |
| 23 | N/A | None | 14.428 ns | data_in[1] | FZ |
| 24 | N/A | None | 14.371 ns | F | data_out[6] |
| 25 | N/A | None | 14.263 ns | data_in[3] | data_out[2] |
| 26 | N/A | None | 14.257 ns | F | data_out[5] |
| 27 | N/A | None | 14.236 ns | data_in[0] | data_out[1] |
| 28 | N/A | None | 14.217 ns | F | FZ |
| 29 | N/A | None | 14.034 ns | FRR | data_out[5] |
| 30 | N/A | None | 14.025 ns | data_in[6] | FZ |
| 31 | N/A | None | 13.996 ns | data_in[2] | data_out[1] |
| 32 | N/A | None | 13.952 ns | FRL | data_out[4] |
| 33 | N/A | None | 13.935 ns | data_in[7] | data_out[7] |
| 34 | N/A | None | 13.907 ns | data_in[1] | data_out[2] |
| 35 | N/A | None | 13.766 ns | data_in[6] | data_out[5] |
| 36 | N/A | None | 13.753 ns | FRR | data_out[1] |
| 37 | N/A | None | 13.645 ns | F | data_out[1] |
| 38 | N/A | None | 13.607 ns | data_in[7] | FC |

## 五、实验结论（实验总结与实验心得）

在本次实验难度较大，在其他同学的帮助下和自己的查阅资料之后，逐步熟练了 QuartusII 的使用，以及易错点，改正了初步程序的错误之后，各个程序和原理图编译成功，紧接着进行功能仿真和时序仿真，结果与理论结果完全符合。再进行模拟机指令译码器的编写，根据实验一实验的文档得到指令译码器功能表，根据功能表，使用 when-else 语言进行编写，在丰富查阅资料和询问学长学姐之后，程序完成，编译成功，再根据程序得到原理图和 RTL 视图。