

实验三 模型机组合部件的实现（二）（实验报告格式案例）

班级 计科 210X 姓名 甘晴void 学号 202108010XXX

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 8 重 3-1 多路复用器。
3. 分析模型机的功能，设计移位逻辑。
4. 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 VERILOG 语言设计模型机的 8 重 3-1 多路复用器；
2. 用 VERILOG 语言设计模型机的移位模块；
3. 用 VERILOG 语言设计模型机的控制信号产生逻辑。

[A]补=2*4-A

三、实验过程

1、8 重 3-1 多路复用器

A) 创建工程（选择的芯片为 family=Cyclone II; name=EP2C5T144C8）

步骤：【File】->【new project wizard】->【next】->【next】->【properties】
->【next】->选择芯片类型 family=Cyclone II, name= EP2C5T144C8->【next】
->【finish】完成工程创建。

When you click Finish, the project will be created with the following settings:

```
Project
directory:
    E:/Quartus II/quartus/mux3_1/

Project
name:      mux3_1
Top-level design entity:  mux3_1

Number of files
added:      0
Number of user libraries
added:      0
Device assignments:

    Family name:      Cyclone II
    Device:           EP2C5T144C8

EDA tools:

    Design
    entry/synthesis:  <None>
    Simulation        ModelSim-Altera (Verilog HDL)
    Timing analysis:  <None>

Operating conditions:

    Core voltage:      1.2V
    Junction temperature
    range:             0-85 度
```

< Back

Next >

Finish

取消





B) 编写源代码

【file】->【Verilog HDL】->写好源代码，保存文件

```
1  module mux3_1(a,b,c,madd,y);
2      input [7:0] a;
3      input [7:0] b;
4      input [7:0] c;
5      input [1:0] madd;
6      output [7:0] y;
7      reg [7:0] y;
8      always@(*)
9      begin
10         case (madd)
11             2'b00:y=a;
12             2'b01:y=b;
13             2'b10:y=c;
14             2'b11:y=8'hZZ;
15             default: ;
16         endcase
17     end
18
19 endmodule
```

C) 编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）

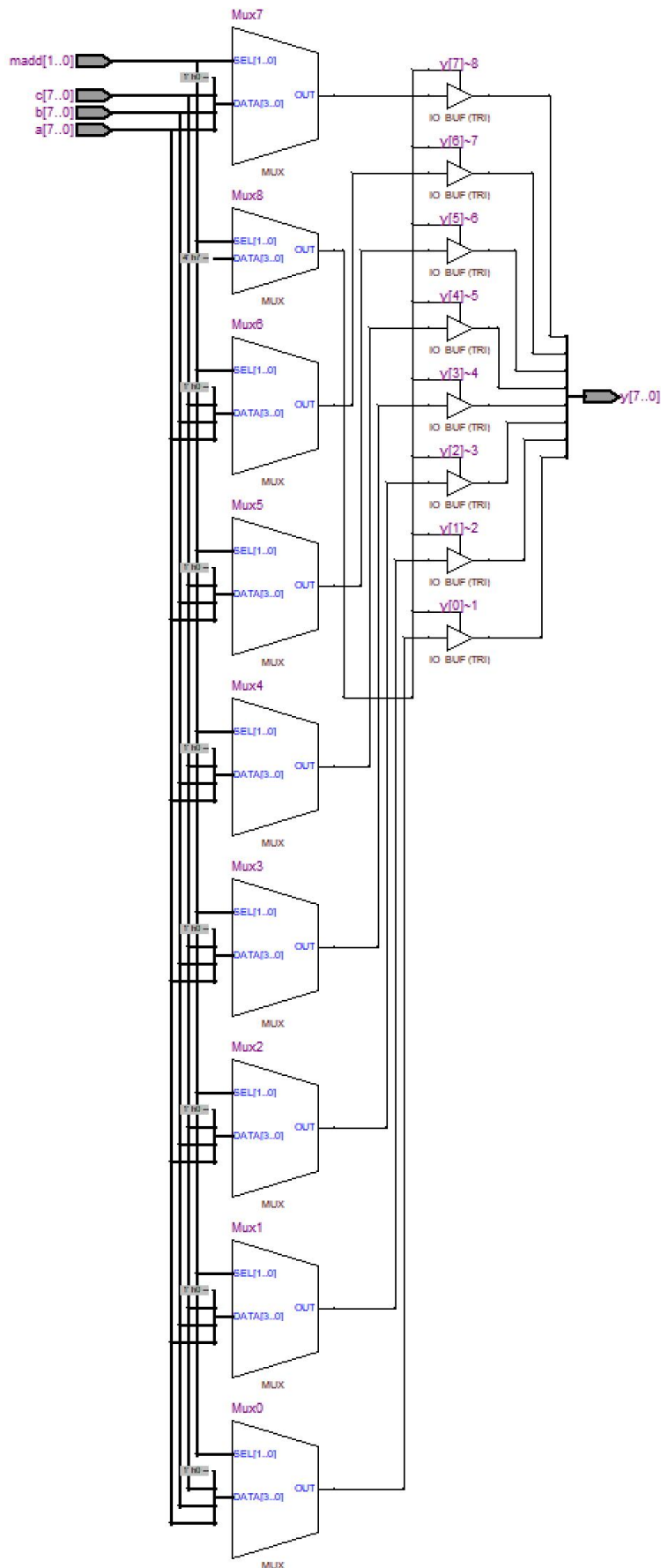
警告信息:

	Warning: Feature LogicLock is not available with your current license
	Warning: No exact pin location assignment(s) for 34 pins of 34 total pins
	Warning: Found 8 output pins without output pin load capacitance assignment
	Warning: The Reserve All Unused Pins setting has not been specified, and will default to 'As output driving ground'.

资源消耗:

Flow Status	Successful - Tue Nov 29 22:17:50 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	mux3_1
Top-level Entity Name	mux3_1
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	17 / 4,608 (< 1 %)
Total combinational functions	17 / 4,608 (< 1 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	34 / 89 (38 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

D) RTL 视图

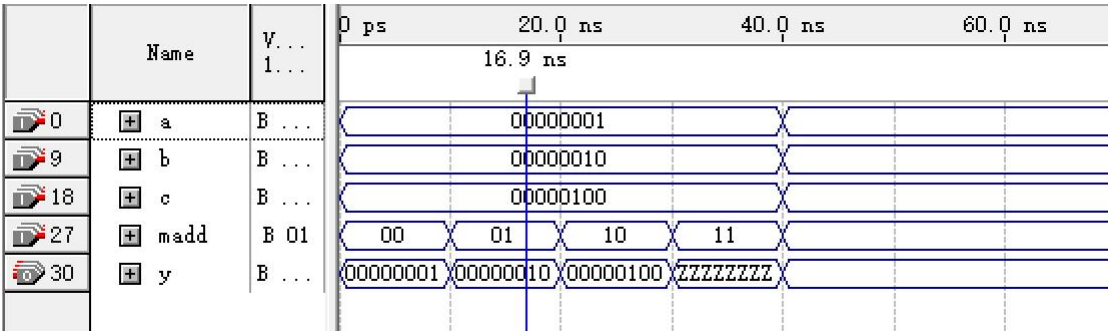


视图分析：

分析：
由视图可得，视图左边为输入，右边为输出。其中连接有一系列的元器件。比如比较器：当输入相等时输出 1，不相等时输出 0；还有大部分的 2-1 选择器构成，当控制信号为 0 时，输出第一位，控制信号为 1 时，输出第二位。图中输入信号为 madd 和 a, b, c, 输出信号为 y。各个输出端口之间通过导线相连。

结论：
一个功能的实现需要经过多重门的处理后才能实现，一个元件的内部原理结构图十分复杂。

E) 功能仿真波形

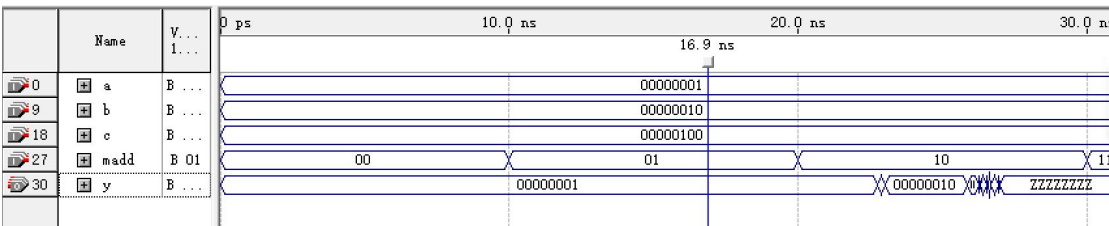


结果分析及结论：

分析：
功能仿真是指不考虑器件延时和布线延时的理想情况下对源代码进行逻辑功能的验证。由仿真波形可得，对于输入状态的变化，输出结果实时变化，没有延迟，其结果与电路设计的真值表的结果相对应。
madd=00 时，控制输出 y 等于 a，正确
madd=01 时，控制输出 y 等于 b，正确
Madd=10 时，控制输出 y 等于 c，正确
Madd=11 时，控制输出高阻态，正确

结论：
功能仿真操作简单，能体现和验证实验的功能，但忽略延迟的影响会使结果与实际结果有一定误差。

F) 时序仿真波形



结果分析及结论：

分析：
时序仿真是指在布线后进行，是最接近真实器件运行的仿真，它与特定的器件有关，又包含了器件和布线的延时信息。由波形可得，当输入状态发生改变时，输出结果并未同时改变，而是有一定延迟，同时由于输入状态的改变，导致电路出现“冒险”，导致输出结果并未与预期结果相同。

结论：
时序仿真可以用来验证程序在目标器件中的时序关系。同时考虑了器件的延迟后，其输出结果跟接近实际情况，但是考虑的情况过多，不容易操作，容易产生错误。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。

G) 时序分析

操作方法是：编译后，在 compilation report 中选择【timing analysis】-【summary】和【tpd】

Timing Analyzer Summary										
	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths	
1	Worst-case tpd	N/A	None	15.082 ns	c[6]	y[6]	--	--	0	
2	Total number of failed paths								0	

tpd						
	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	15.082 ns	c[6]	y[6]	
2	N/A	None	15.027 ns	a[6]	y[6]	
3	N/A	None	13.820 ns	a[7]	y[7]	
4	N/A	None	13.595 ns	madd[0]	y[3]	
5	N/A	None	13.437 ns	madd[0]	y[7]	
6	N/A	None	13.254 ns	c[7]	y[7]	
7	N/A	None	13.191 ns	c[3]	y[3]	
8	N/A	None	13.104 ns	a[5]	y[5]	
9	N/A	None	13.101 ns	madd[0]	y[6]	
10	N/A	None	13.046 ns	madd[0]	y[0]	
11	N/A	None	13.030 ns	madd[0]	y[5]	
12	N/A	None	12.962 ns	c[5]	y[5]	
13	N/A	None	12.852 ns	b[7]	y[7]	
14	N/A	None	12.840 ns	c[2]	y[2]	
15	N/A	None	12.737 ns	madd[0]	y[1]	
16	N/A	None	12.721 ns	madd[0]	y[4]	
17	N/A	None	12.714 ns	a[1]	y[1]	
18	N/A	None	12.695 ns	b[5]	y[5]	
19	N/A	None	12.674 ns	a[2]	y[2]	
20	N/A	None	12.403 ns	a[4]	y[4]	
21	N/A	None	12.388 ns	a[3]	y[3]	
22	N/A	None	12.330 ns	b[3]	y[3]	
23	N/A	None	12.313 ns	madd[0]	y[2]	
24	N/A	None	12.007 ns	b[4]	y[4]	
25	N/A	None	11.875 ns	b[6]	y[6]	
26	N/A	None	11.758 ns	c[4]	y[4]	
27	N/A	None	11.689 ns	b[1]	y[1]	
28	N/A	None	11.080 ns	c[1]	y[1]	
29	N/A	None	10.581 ns	b[2]	y[2]	
30	N/A	None	10.105 ns	madd[1]	y[6]	
31	N/A	None	9.077 ns	madd[1]	y[3]	
32	N/A	None	8.914 ns	madd[1]	y[7]	
33	N/A	None	8.489 ns	a[0]	y[0]	
34	N/A	None	8.474 ns	madd[1]	y[5]	
35	N/A	None	8.420 ns	c[0]	y[0]	
36	N/A	None	8.105 ns	madd[1]	y[0]	
37	N/A	None	7.852 ns	madd[1]	y[2]	
38	N/A	None	7.836 ns	b[0]	y[0]	
39	N/A	None	7.772 ns	madd[1]	y[4]	
40	N/A	None	7.765 ns	madd[1]	y[1]	

结果分析及结论：

分析：

由图可得，Timing Analyzer Summmary 总结所有经典定时分析的结果，并报告每个定时特性的最坏情况定时。比如从 b[7]到 y[7]的最坏定时情况的 tpd 为 12.852ns。下面的 tpd 报告表则给出了源节点和目标节点之间的 tpd 延迟间，比如第二行中 a[7]到 y[7] 的 tpd 为 13.820ns。

结论：

实际连接图中个元器件连接之间是存在时间延迟的，而且不同的元器件之间的时间延迟也不相同。

2、移位逻辑

A) 创建工程（选择的芯片为 family=FLEX10K; name=EPF10K20T1144-4）

New Project Wizard: Summary [page 5 of 5]

When you click Finish, the project will be created with the following settings:

Project directory:
E:/Quartus II/quartus/shift/

Project name:
shift

Top-level design entity:
shift

Number of files added:
0

Number of user libraries added:
0

Device assignments:

Family name:
FLEX10K

Device:
EPF10K20T1144-4

EDA tools:

Design entry/synthesis:
<None>

Simulation
ModelSim-Altera (Verilog HDL)

Timing analysis:
<None>

Operating conditions:

Core voltage:
5.0V

Junction temperature range:
0-85 度

< Back

Next >

Finish

取消

第 8 页 共 22 页

B) 编写源代码

```

1  module shift(fbus,flbus,frbus,a,w,cf);
2      input fbus,flbus,frbus;
3      input [7:0] a;
4      output [7:0] w;
5      output cf;
6      reg [7:0] w;
7      reg cf;
8      always@(*)
9      begin
10         if (fbus==1 && flbus==0 && frbus==0)
11         begin
12             w=a;
13             cf=0;
14         end
15         else if (fbus==0 && flbus==1 && frbus==0)
16         begin
17             w={a[6:0],a[7]};
18             cf=a[7];
19         end
20         else if (fbus==0 && flbus==0 && frbus==1)
21         begin
22             w={a[0],a[7:1]};
23             cf=a[0];
24         end
25         else
26         begin
27             w=8'hZZ;
28             cf=0;
29         end
30     end
31
32
33 endmodule

```

C) 编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）

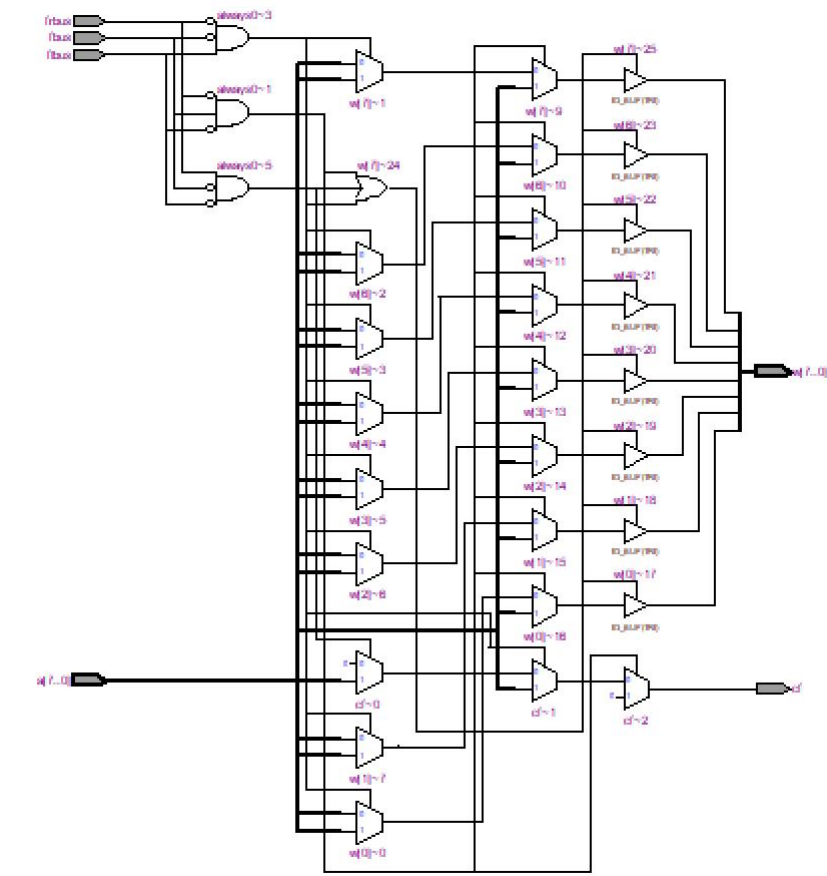
警告信息：

无警告信息

资源消耗：

Flow Status	Successful - Tue Nov 29 23:13:14 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	shift
Top-level Entity Name	shift
Family	FLEX10K
Device	EPF10K20TI144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	21 / 1,152 (2 %)
Total pins	20 / 102 (20 %)
Total memory bits	0 / 12,288 (0 %)

D) RTL 视图

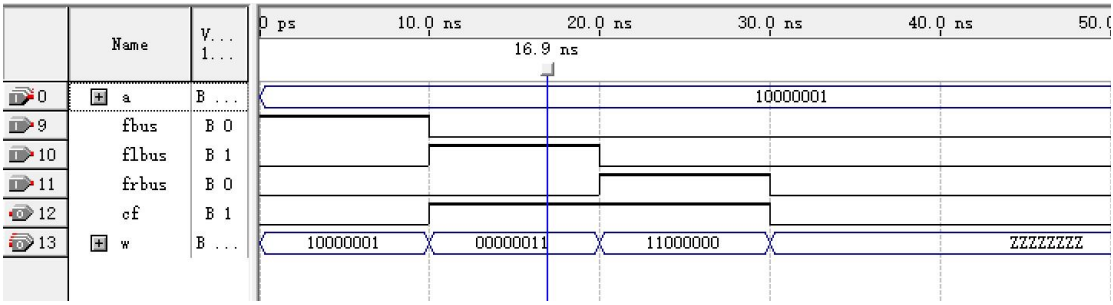


结果分析：

分析：
由视图可得，视图左边为输入，右边为输出。其中连接有一系列的元器件。比如比较器：当输入相等时输出 1，不相等时输出 0；还有大部分的 2-1 选择器构成，当控制信号为 0 时，输出第一位，控制信号为 1 时，输出第二位。图中输入信号为 fbus, frbus, flbus 和 a，输出信号为 w。各个输出端口之间通过导线相连。

结论：
一个功能的实现需要经过多重门的处理后才能实现，一个元件的内部原理结构图十分复杂。

E) 功能仿真波形



结果分析及结论：

分析：

功能仿真是指不考虑器件延时和布线延时的理想情况下对源代码进行逻辑功能的验证。由仿真波形可得，对于输入状态的变化，输出结果实时变化，没有延迟，其结果与电路设计的真值表的结果相对应。

当 fbus=1, frbus=0, flbus=0, 不执行移位操作，输出等于输入，cf 不改变

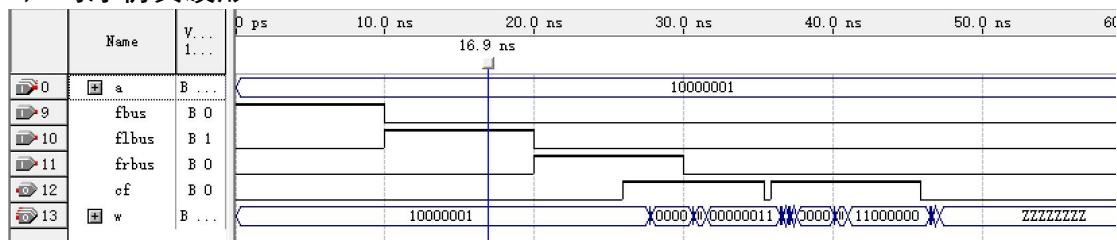
当 fbus=0, frbus=1, flbus=0, 执行右移，输出等于输入右移移位，有进位的话 cf 为 1

当 fbus=0, frbus=0, flbus=1, 执行左移，输出等于输入左移一位，cf 不改变
当控制信号全为 0 时，输出为高阻态，正确

结论：

功能仿真操作简单，能体现和验证实验的功能，但忽略延迟的影响会使结果与实际结果有一定误差。

F) 时序仿真波形



结果分析及结论：

分析：

时序仿真是指在布线后进行，是最接近真实器件运行的仿真，它与特定的器件有关，又包含了器件和布线的延时信息。由波形可得，当输入状态发生改变时，输出结果并未同时改变，而是有一定延迟，同时由于输入状态的改变，导致电路出现“冒险”，导致输出结果并未与预期结果相同。

结论：

时序仿真可以用来验证程序在目标器件中的时序关系。同时考虑了器件的延迟后，其输出结果跟接近实际情况，但是考虑的情况过多，不容易操作，容易产生错误。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。

G) 时序分析

Timing Analyzer Summary										
	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths	
1	Worst-case tpd	N/A	None	21.700 ns	fbus	w[5]	--	--	0	
2	Total number of failed paths								0	

tpd						
	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	21.700 ns	flbus	w[5]	
2	N/A	None	21.700 ns	fbus	w[5]	
3	N/A	None	21.500 ns	flbus	w[1]	
4	N/A	None	21.500 ns	fbus	w[1]	
5	N/A	None	21.200 ns	frbus	w[5]	
6	N/A	None	21.000 ns	frbus	w[1]	
7	N/A	None	20.900 ns	flbus	w[3]	
8	N/A	None	20.900 ns	fbus	w[3]	
9	N/A	None	20.700 ns	flbus	w[7]	
10	N/A	None	20.700 ns	fbus	w[7]	
11	N/A	None	20.700 ns	flbus	w[2]	
12	N/A	None	20.700 ns	fbus	w[2]	
13	N/A	None	20.700 ns	flbus	w[0]	
14	N/A	None	20.700 ns	fbus	w[0]	
15	N/A	None	20.600 ns	flbus	w[6]	
16	N/A	None	20.600 ns	fbus	w[6]	
17	N/A	None	20.600 ns	flbus	w[4]	
18	N/A	None	20.600 ns	fbus	w[4]	
19	N/A	None	20.400 ns	frbus	w[3]	
20	N/A	None	20.200 ns	frbus	w[7]	
21	N/A	None	20.200 ns	frbus	w[2]	
22	N/A	None	20.200 ns	frbus	w[0]	
23	N/A	None	20.100 ns	frbus	w[6]	
24	N/A	None	20.100 ns	frbus	w[4]	
25	N/A	None	20.000 ns	a[4]	w[5]	
26	N/A	None	19.400 ns	a[1]	w[2]	
27	N/A	None	19.400 ns	a[1]	w[0]	
28	N/A	None	19.200 ns	a[4]	w[3]	
29	N/A	None	19.100 ns	a[5]	w[6]	
30	N/A	None	19.100 ns	a[5]	w[4]	
31	N/A	None	19.000 ns	a[6]	w[5]	
32	N/A	None	18.500 ns	a[6]	w[7]	
33	N/A	None	18.500 ns	a[3]	w[2]	
34	N/A	None	17.900 ns	a[3]	w[4]	
35	N/A	None	17.300 ns	a[5]	w[5]	
36	N/A	None	17.300 ns	a[1]	w[1]	
37	N/A	None	17.200 ns	a[0]	w[1]	
35	N/A	None	17.100 ns	a[2]	w[1]	
39	N/A	None	16.400 ns	a[0]	w[7]	
40	N/A	None	16.400 ns	a[7]	w[0]	
41	N/A	None	16.300 ns	a[2]	w[3]	
42	N/A	None	16.100 ns	a[0]	cf	
43	N/A	None	16.000 ns	a[7]	w[6]	
44	N/A	None	16.000 ns	a[4]	w[4]	
45	N/A	None	15.900 ns	frbus	cf	
46	N/A	None	15.900 ns	flbus	cf	
47	N/A	None	15.900 ns	fbus	cf	
48	N/A	None	15.300 ns	a[3]	w[3]	
49	N/A	None	15.000 ns	a[6]	w[6]	
50	N/A	None	13.500 ns	a[7]	w[7]	
51	N/A	None	13.500 ns	a[0]	w[0]	
52	N/A	None	13.400 ns	a[2]	w[2]	
53	N/A	None	12.700 ns	a[7]	cf	

结果分析及结论：

分析：

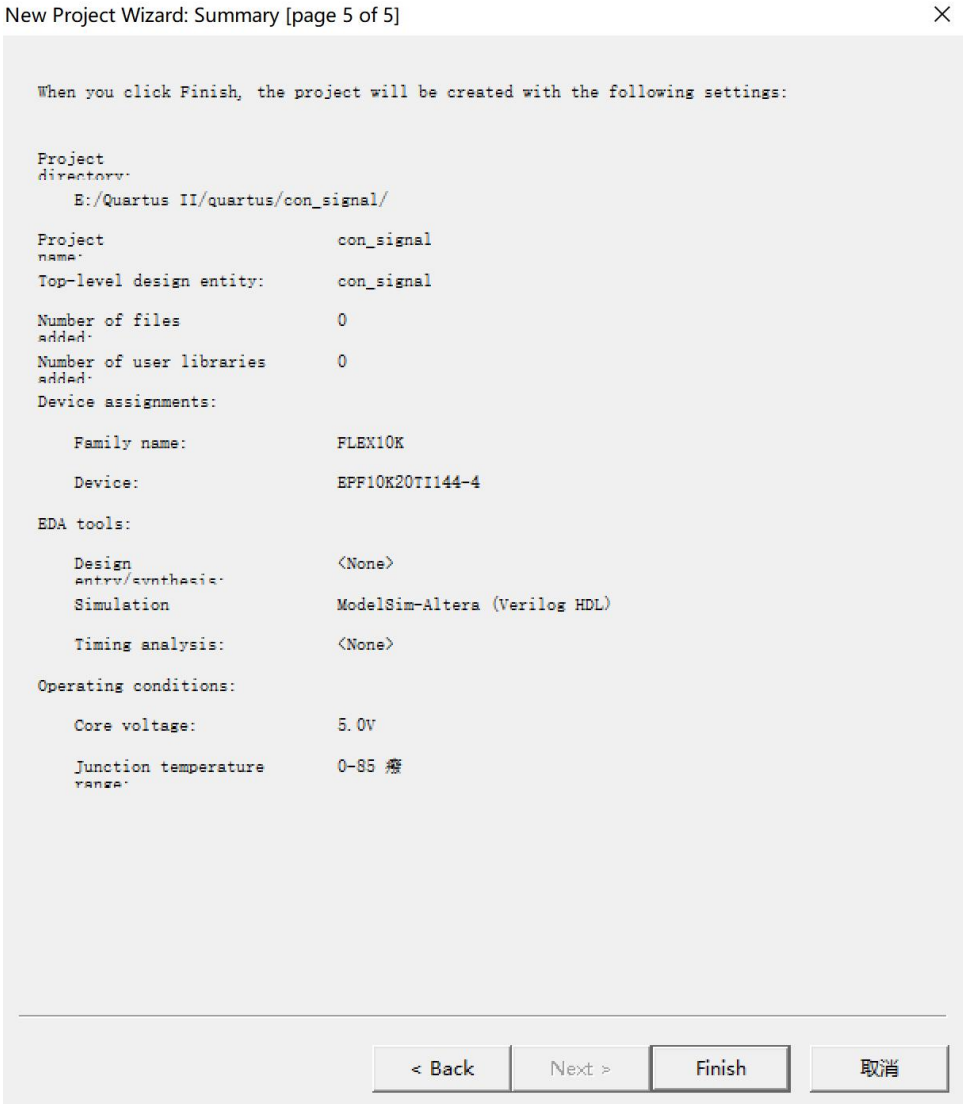
由图可得，Timing Analyzer Summary 总结所有经典定时分析的结果，并报告每个定时特性的最坏情况定时。比如从 fbus 到 w[0] 的最坏定时情况的 tpd 为 20.700ns。下面的 tpd 报告表则给出了源节点和目标节点之间的 tpd 延迟时间，比如第二行中 fbus 到 w[7] 的 tpd 为 20.700ns。

结论：

实际连接图中个元器件连接之间是存在时间延迟的，而且不同的元器件之间的时间延迟也不相同。

3、控制信号产生逻辑

A) 创建工程（选择的芯片为 family=FLEX10K; name=EPF10K20T1144-4）



B) 编写源代码

```

1 module con_signal(mova,movb,movc,add,sub,and1,not1,rsr,rs1,jmp,jz,z,jc,c,in1,out1,nop,halt,ir,sm,reg_ra,reg_wa,madd,
  pc_ld,pc_inc,reg_we,ram_xl,ram_dl,alu_m,shi_fbus,shi_flbus,shi_frbus,ir_ld,cf_en,zf_en,sm_en,in_en,out_en,reg_ra,
  reg_wa,madd,alu_s);
2   input mova,movb,movc,add,sub,and1,not1,rsr,rs1,jmp,jz,z,jc,c,in1,out1,nop,halt,sm;
3   input [7:0] ir;
4   output [1:0] reg_ra,reg_wa,madd;
5   output [3:0] alu_s;
6   output pc_ld,pc_inc,reg_we,ram_xl,ram_dl,alu_m,shi_fbus,shi_flbus,shi_frbus,ir_ld,cf_en,zf_en,sm_en,in_en,out_en;
7   reg pc_ld,pc_inc,reg_we,ram_xl,ram_dl,alu_m,shi_fbus,shi_flbus,shi_frbus,ir_ld,cf_en,zf_en,sm_en,in_en,out_en;
8   reg [1:0] reg_ra,reg_wa,madd;
9   reg [3:0] alu_s;
10  always@(*)
11  begin
12      //sm翻转控制
13      sm_en=!halt;
14
15      //ALU
16      alu_m=add || sub || and1 || not1 || rs1 || rsr || out1;
17      alu_s[3:0]=ir[7:4];
18      cf_en=add || sub || rsr || rs1;
19      zf_en=add || sub;
20
21      //移位逻辑
22      shi_fbus=mova || movb || add || sub || and1 || not1 || out1;
23      shi_frbus=rsr;
24      shi_flbus=rs1;
25
26      //RAM
27      ram_xl=movb;
28      ram_dl=movc || jmp || ( z && jz ) || ( c && jc ) || !sm;
29
30      //指令寄存器
31      ir_ld=!sm;
32
33      //寄存器组
34      reg_ra=ir[1:0];
35      reg_wa=ir[3:2];
36      reg_we=movb || jmp || ( z && jz ) || ( c && jc ) || out1 || !sm;
37
38      //计数器PC
39      pc_ld=jmp || ( z && jz ) || ( c && jc );
40      pc_inc=( !z && jz ) || ( !c && jc ) || !sm;
41
42      //选择器MADD
43      if( movb && sm ) madd=2'b10;
44      else if( movc && sm ) madd=2'b01;
45      else if( !sm ) madd=2'b00;
46      else madd=2'b00;
47
48      //输入输出设备
49      in_en=in1;
50      out_en=out1;
51
52  end
53 endmodule

```

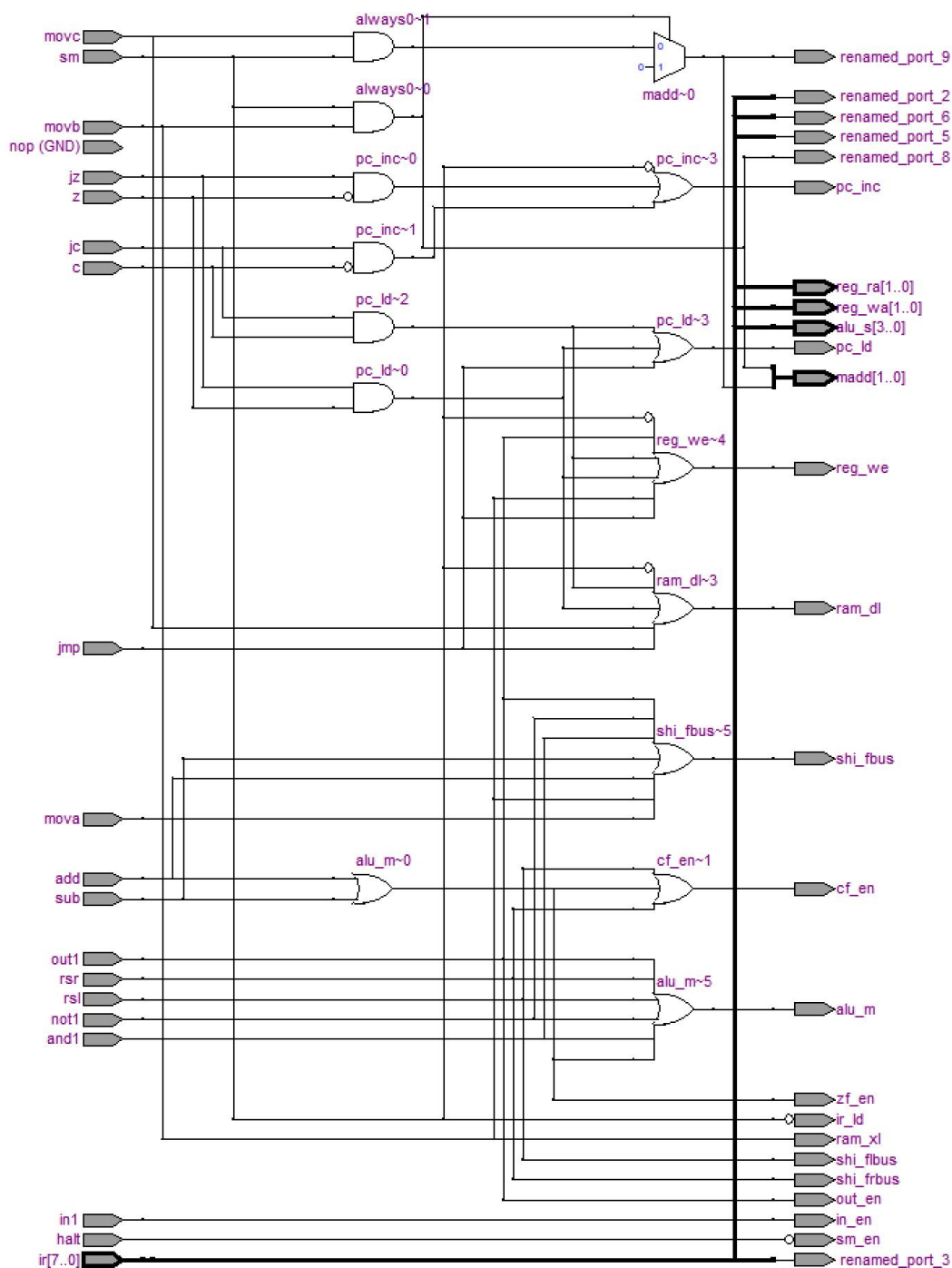
D) 编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）
警告信息：

Warning: Design contains 1 input pin(s) that do not drive logic

资源消耗：

Warning (10136): Verilog HDL Module Declaration warning at con_signal.v(1): port "reg_ra" already exists in the list of ports
 Warning (10136): Verilog HDL Module Declaration warning at con_signal.v(1): port "reg_wa" already exists in the list of ports
 Warning (10136): Verilog HDL Module Declaration warning at con_signal.v(1): port "madd" already exists in the list of ports
 Warning: Design contains 1 input pin(s) that do not drive logic

D) RTL 视图



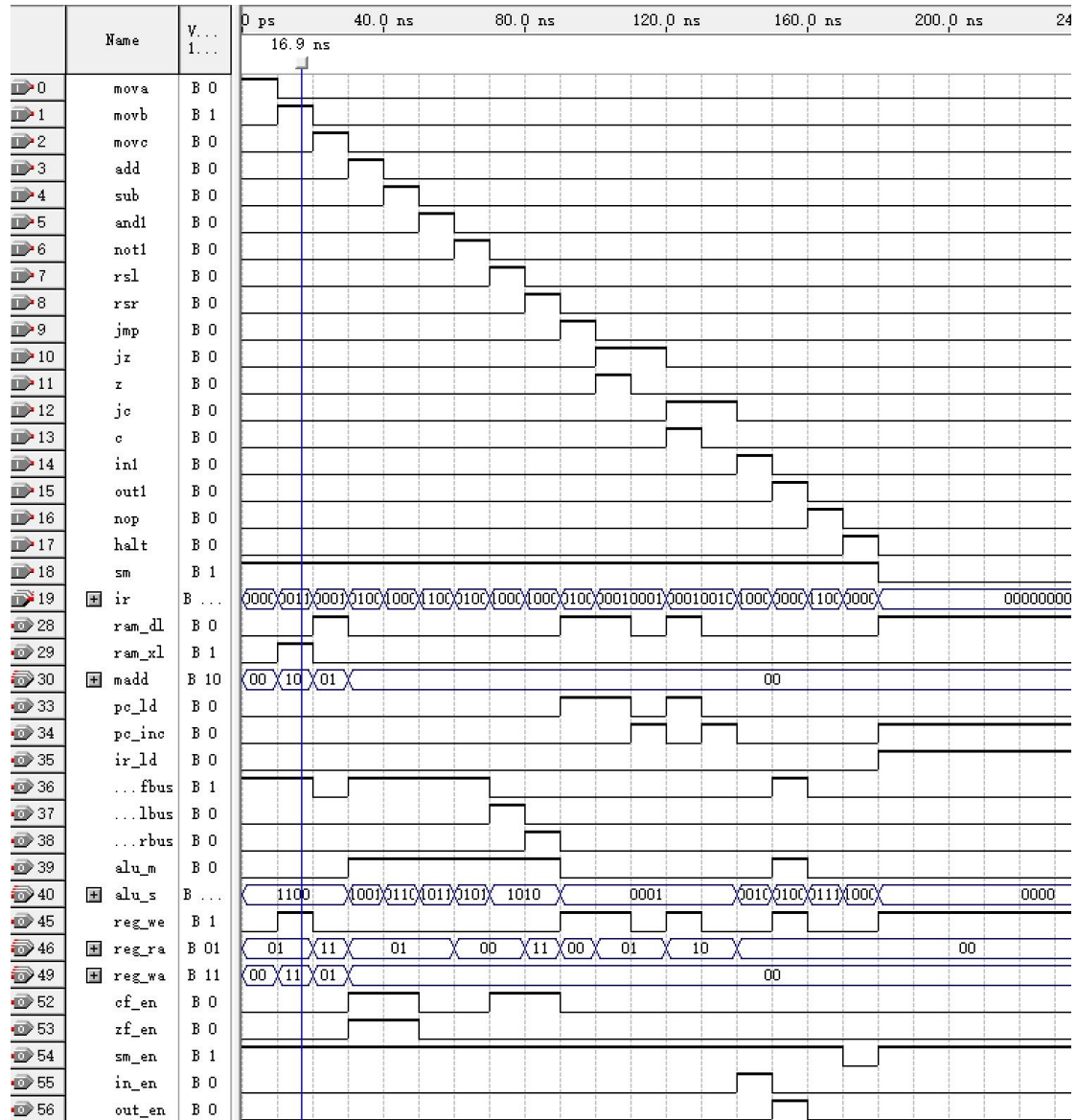
结果分析:

由视图可得，视图左边为输入，右边为输出。其中连接有一系列的元器件。比如比较器：当输入相等时输出 1，不相等时输出 0；还有大部分的与或门。图中输入信号为 sm 等 20 个，输出信号包括 reg_ra 等 19 种情况。各个输出端口之间通过导线相连。

结论:

一个功能的实现需要经过多重门的处理后才能实现，一个元件的内部原理结构图十分复杂。

E) 功能仿真波形



结果分析及结论:

分析:

功能仿真是指不考虑器件延时和布线延时的理想情况下对源代码进行逻辑功能的验证。由仿真波形可得，对于输入状态的变化，输出结果实时变化，没有延迟，其结果与电路设计的真值表的结果相对应。

(1) 当 mova 指令执行时，shi_fbus 和 sm_en 输出 1，其他输出为 0，madd 输出 00，alu_s 输出为 1100，reg_ra 输出 01，reg_wa 输出 00，正确

(2) 当 movb 指令执行时，ram_xl 和 shi_fbus 和 reg_we 和 sm_en 输出为 1，其他输出为 0，madd 输出为 10，alu_s 输出为 1100，reg_ra 输出 01，reg_wa 输出 11，正确

(3) 当 movc 指令执行时，ram_dl 和 sm_en 输出为 1，其他输出为 0，madd 输出 01，alu_s 输出 1100，reg_ra 输出 11，reg_wa 输出 01，正确

(4) 当 add 指令执行时，shi_fbus，alu_en，cf_en，zf_en，sm_en 输出为 1，其他输出

为 0, alu_s 为 1001, reg_ra 输出 01, reg_wa 输出 00, 正确

(5) 当 sub 指令执行时, shi_fbus 和 alu_m, cf_en, zf_en 和 sm_en 输出为 1, 其他输出为 0, alu_s 输出 0110, reg_ra 输出 01, reg_wa 输出 00, 正确

(6) 当 andl 指令执行时, shi_fbus 和 alu_m 和 sm_en 输出 1, 其他输出 0, alu_s 输出 1011, reg_ra 输出 01, reg_wa 输出 00, 正确

(7) notl 指令执行时, shi_fbus 和 alu_m 和 sm_en 输出 1, 其他输出 0, alu_s 输出 0101, reg_ra 输出 00, reg_wa 输出 00, 正确

(8) rsl 指令执行时, shi_fbus 和 alu_m 和 cf_en 和 sm_en 输出 0, 其他输出 0, alu_s 输出 1010, reg_ra 和 reg_wa 输出 00, 正确

(9) rsr 指令执行时, shi_fbus 和 alu_m 和 cf_en 和 sm_en 输出 1, 其他输出 0, alu_s 输出 10110, reg_ra 输出 11, reg_wa 输出 00, 正确

(10) jmp 指令执行时, ram_dl, pc_ld, reg_we 和 sm_en 输出 1, 其他输出 0, alu_s 输出 0001, reg_ra 和 reg_wa 输出 00, 正确

(11) jz 指令为 1 和 jc 指令为 1 时, 若 z 和 c 为 1 时, ram_dl 和 pc_ld 和 reg_we 和 sm_en 输出为 1, 其他输出为 0, 正确

若 z 和 c 为 0 时, pc_inc 和 reg_we 和 sm_en 输出 1, 其他输出 0, 正确

(12) inl 指令执行时, sm_en 和 in_en 输出 1, 其他输出 0, 正确

(13) outl 指令执行时, sm_en 和 out_en 输出 1, 其他输出 0, 正确

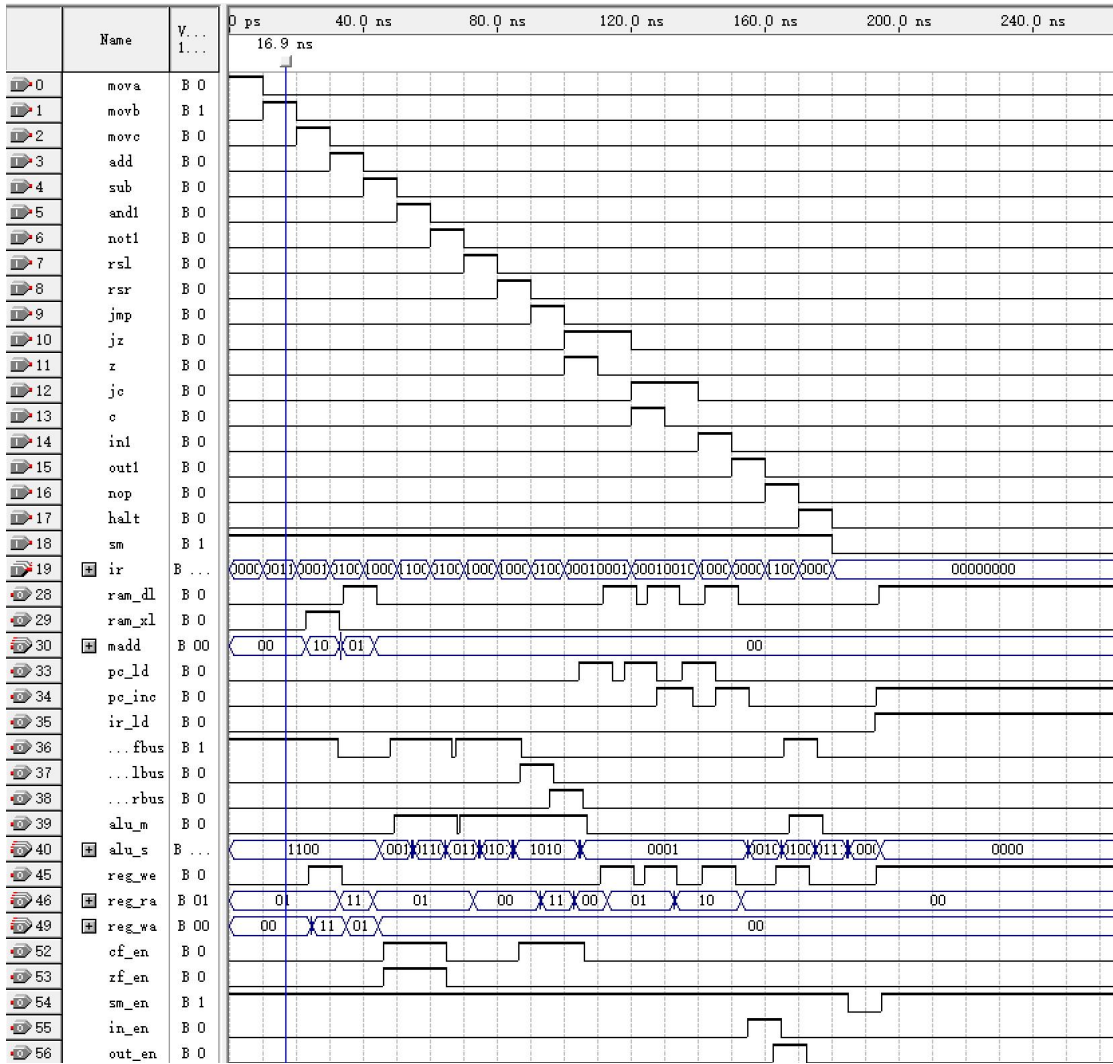
(14) nop 指令执行时, sm_en 输出 1, 其他输出 0, 正确

(15) halt 指令执行时, 输出全为 0, 正确

结论:

功能仿真操作简单, 能体现和验证实验的功能, 但忽略延迟的影响会使结果与实际结果有一定误差。0, reg_ra 输出 11, reg_wa 输出 00, 正确

F) 时序仿真波形



分析：

时序仿真是指在布线后进行，是最接近真实器件运行的仿真，它与特定的器件有关，又包含了器件和布线的延时信息。由波形可得，当输入状态发生改变时，输出结果并未同时改变，而是有一定延迟，同时由于输入状态的改变，导致电路出现“冒险”，导致输出结果并未与预期结果相同。

结论：

时序仿真可以用来验证程序在目标器件中的时序关系。同时考虑了器件的延迟后，其输出结果跟接近实际情况，但是考虑的情况过多，不容易操作，容易产生错误。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。

G) 时序分析

实验三 模型机组部件的实现（二）

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	24.900 ns	jz	ram_d1	--	--	0
2	Total number of failed paths								0

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	24.900 ns	jz	ram_d1
2	N/A	None	24.400 ns	z	ram_d1
3	N/A	None	24.100 ns	jz	reg_we
4	N/A	None	23.600 ns	z	reg_we
5	N/A	None	22.000 ns	jc	ram_d1
6	N/A	None	22.000 ns	c	ram_d1
7	N/A	None	21.500 ns	jmp	ram_d1
8	N/A	None	21.200 ns	jc	reg_we
9	N/A	None	21.200 ns	c	reg_we
10	N/A	None	20.700 ns	jmp	reg_we
11	N/A	None	19.400 ns	add	alu_m
12	N/A	None	19.100 ns	and1	alu_m
13	N/A	None	18.600 ns	not1	alu_m
14	N/A	None	18.300 ns	jz	pc_inc
15	N/A	None	18.100 ns	sub	alu_m
16	N/A	None	18.000 ns	add	shi_fbus
17	N/A	None	18.000 ns	jz	pc_ld
18	N/A	None	17.800 ns	z	pc_inc
19	N/A	None	17.700 ns	and1	shi_fbus
20	N/A	None	17.500 ns	z	pc_ld
21	N/A	None	17.300 ns	rsl	alu_m
22	N/A	None	17.200 ns	not1	shi_fbus
23	N/A	None	17.100 ns	rsl	shi_fbus
24	N/A	None	17.100 ns	out1	alu_m
25	N/A	None	16.900 ns	rsr	alu_m
26	N/A	None	16.700 ns	sub	shi_fbus
27	N/A	None	16.500 ns	rsl	cf_en
28	N/A	None	16.200 ns	add	cf_en
29	N/A	None	16.100 ns	add	zf_en
30	N/A	None	16.100 ns	rsr	cf_en
31	N/A	None	16.100 ns	movb	shi_fbus
32	N/A	None	15.700 ns	out1	shi_fbus
33	N/A	None	15.600 ns	rsr	shi_fbus
34	N/A	None	15.500 ns	ir[2]	renamed_port_6
35	N/A	None	15.400 ns	jc	pc_inc
36	N/A	None	15.400 ns	c	pc_inc
37	N/A	None	15.100 ns	jc	pc_ld
38	N/A	None	15.100 ns	c	pc_ld
39	N/A	None	14.900 ns	ir[6]	alu_s[2]
40	N/A	None	14.900 ns	ir[4]	alu_s[0]
41	N/A	None	14.900 ns	ir[3]	reg_wa[1]
42	N/A	None	14.900 ns	in1	in_en
43	N/A	None	14.900 ns	halt	sm_en
44	N/A	None	14.900 ns	sub	cf_en
45	N/A	None	14.900 ns	ir[3]	renamed_port_5
46	N/A	None	14.800 ns	ir[5]	alu_s[1]
47	N/A	None	14.800 ns	sub	zf_en
48	N/A	None	14.600 ns	ir[7]	alu_s[3]
49	N/A	None	14.600 ns	ir[2]	reg_wa[0]
50	N/A	None	14.600 ns	jmp	pc_ld
51	N/A	None	14.400 ns	sm	renamed_port_8
52	N/A	None	14.400 ns	movb	renamed_port_9
53	N/A	None	14.200 ns	movc	ram_d1
54	N/A	None	14.200 ns	movc	renamed_port_9
55	N/A	None	13.900 ns	sm	ram_d1
56	N/A	None	13.900 ns	movb	renamed_port_8
57	N/A	None	13.900 ns	sm	renamed_port_9
58	N/A	None	13.600 ns	movb	madd[0]
59	N/A	None	13.600 ns	movb	reg_we
60	N/A	None	13.400 ns	sm	madd[1]
61	N/A	None	13.400 ns	movc	madd[0]
62	N/A	None	13.400 ns	out1	reg_we
63	N/A	None	13.100 ns	movb	madd[1]
64	N/A	None	13.100 ns	sm	madd[0]
65	N/A	None	13.100 ns	movb	ram_x1

66	N/A	None	13.100 ns	sm	reg_we
67	N/A	None	13.100 ns	sm	pc_inc
68	N/A	None	13.000 ns	ir[1]	reg_ra[1]
69	N/A	None	13.000 ns	ir[1]	renamed_port_2
70	N/A	None	12.900 ns	ir[0]	reg_ra[0]
71	N/A	None	12.900 ns	ir[0]	renamed_port_3
72	N/A	None	12.800 ns	sm	ir_ld
73	N/A	None	12.700 ns	movb	shi_fbus
74	N/A	None	12.400 ns	outl	out_en

结果分析及结论：

分析：

由图可得，Timing Analyzer Summary 总结所有经典定时分析的结果，并报告每个定时特性的最坏情况定时。比如从 sub 到 reg_we 的最坏定时情况的 tpd 为 26.600ns。下面的 tpd 报告表则给出了源节点和目标节点之间的 tpd 延迟时间，比如第二行中 and1 到 reg_we 的 tpd 为 26.300ns。

结论：

实际连接图中个元器件连接之间是存在时间延迟的，而且不同的元器件之间的时间延迟也不相同。

四、思考题

1. 移位逻辑不工作时，输出应该为何值？为什么？

答：输出应成高阻，防止数据通路发生冲突，产生主线竞争。

2. 移位逻辑的输出 Cf 应该如何处理？

答：

当 fbus=1, frbus=0, flbus=0, 不执行移位操作，输出等于输入，cf 不改变；

当 fbus=0, frbus=1, flbus=0, 执行右移，输出等于输入右移移位，有进位的话 cf 为 1；

当 fbus=0, frbus=0, flbus=1, 执行左移，输出等于输入左移一位，cf 不改变
当控制信号全为 0 时，输出为高阻态；

3. 如何产生正确的控制信号以及具体的编程实现？

答：应当逐个分析每个控制信号在不同的指令下对应的状态，利用逻辑函数进行状态的总和。

五、实验总结、必得体会及建议

1、从需要掌握的理论、遇到的困难、解决的办法以及经验教训等方面进行总结。

（1）需要掌握的理论：基本了解了简易模型机的内部结构和工作原理。同时熟悉了选择器，移位逻辑，控制器的工作原理。学会使用 Verilog 语言编写电路。

（2）遇到的困难：对于 QuartusII 的使用还不够熟练，特别是进行波形仿真的功能仿真和时序仿真分别怎么操作的方面有一定不足。

（3）解决方法：通过上网查询相关资料和询问同学后得以解决问题，并通过分析报告发现电路中的问题。有不理解的还请教了老师，不仅收获了方法还掌握的技巧。

（4）经验教训：对于电子电路的学习一定要肯动手，光是看是学不会的，一定要落到实处，多自己使用软件进行仿真，才能加深对于这门课程的理解。

2、对本实验内容、过程和方法的改进建议（可选项）。

控制器各个控制信号的输出功能表十分重要，可以先引导同学们列出这个对应表。