

实验三 模型机组合部件的实现（二）

班级 信安2101 姓名 杨华 学号 202108060121

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。

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。

New Project Wizard: Summary [page 5 of 5]

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

Project directory:
D:/qua/quaxunlian/11/

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: <None>

Timing analysis: <None>

Operating conditions:

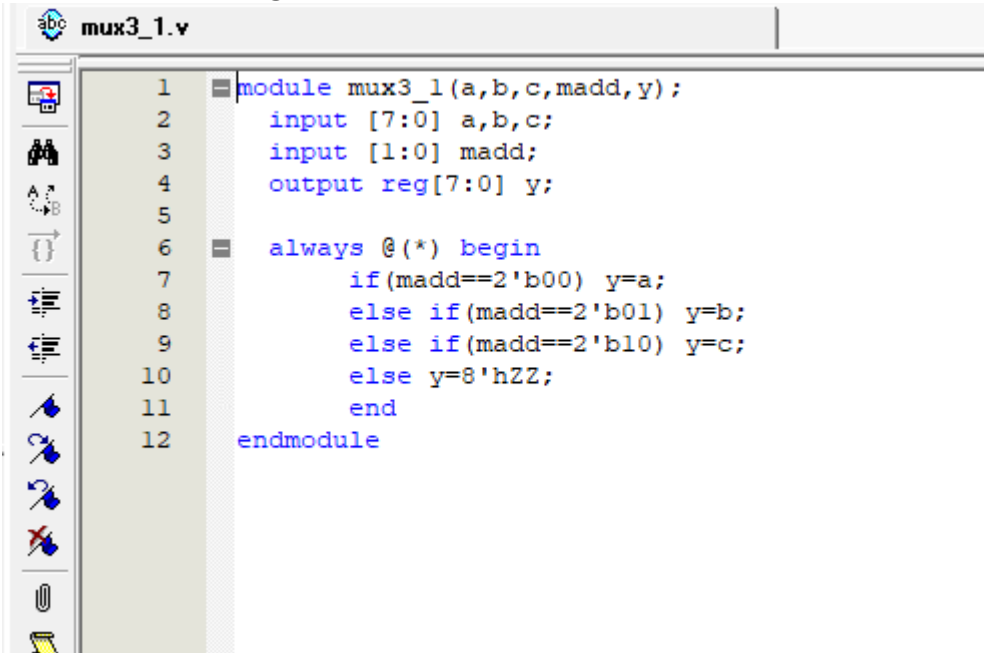
Core voltage: 1.2V

Junction temperature range: 0-85 °C

< 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,b,c;
3     input [1:0] madd;
4     output reg[7:0] y;
5
6     always @(*) begin
7         if(madd==2'b00) y=a;
8         else if(madd==2'b01) y=b;
9         else if(madd==2'b10) y=c;
10        else y=8'hZZ;
11    end
12 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)

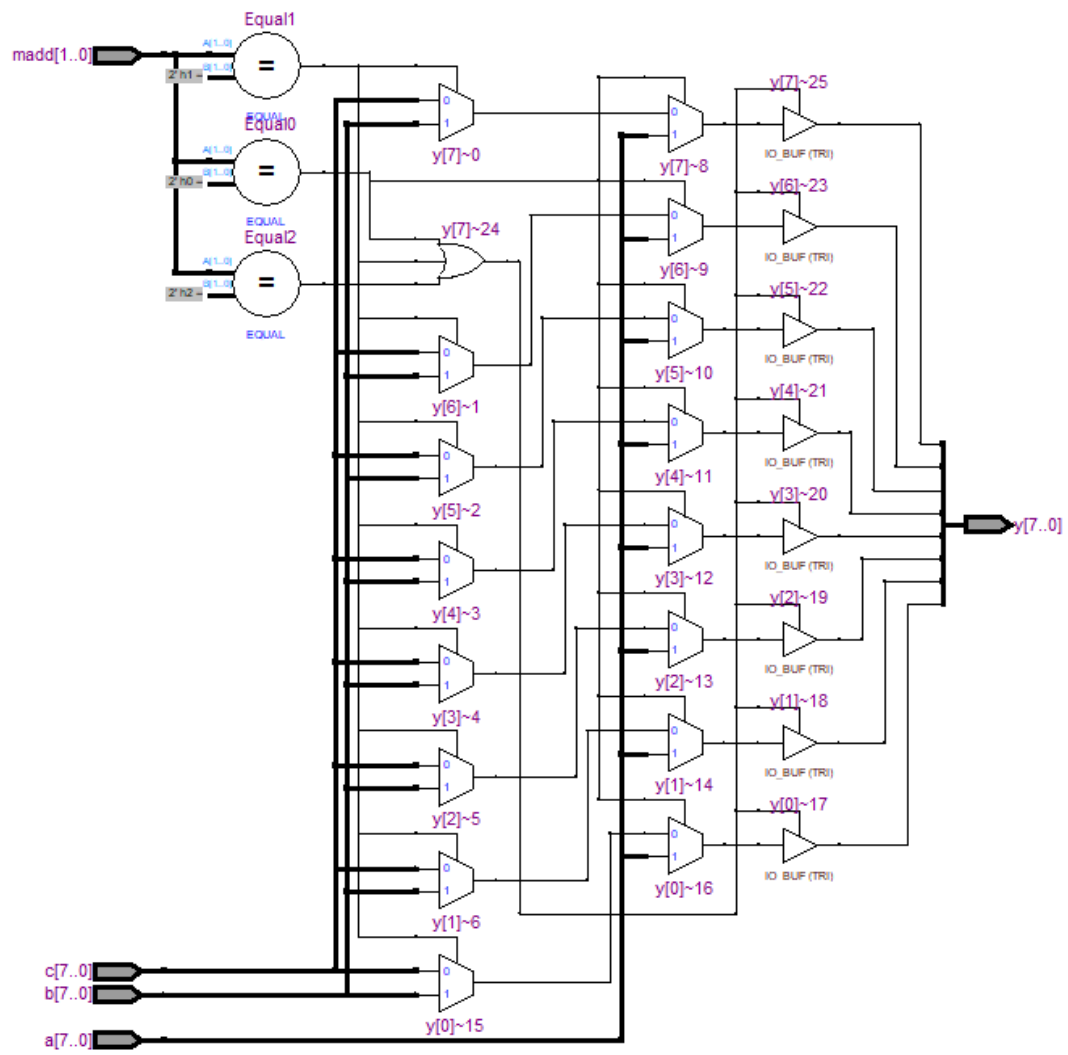
1 警告

 Warning: Feature SignalProbe is not available with your current license

资源消耗:

Flow Status	Successful - Tue Nov 29 17:16:38 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	FLEX10K
Device	EPF10K10TI144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	17 / 576 (3 %)
Total pins	34 / 102 (33 %)
Total memory bits	0 / 6,144 (0 %)

D) RTL 视图



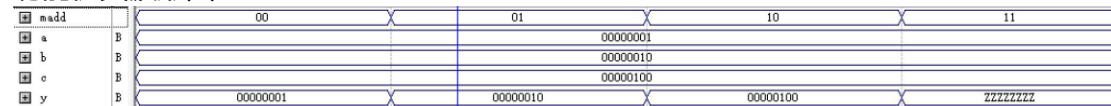
视图分析及结论：

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

功能仿真波形图：



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

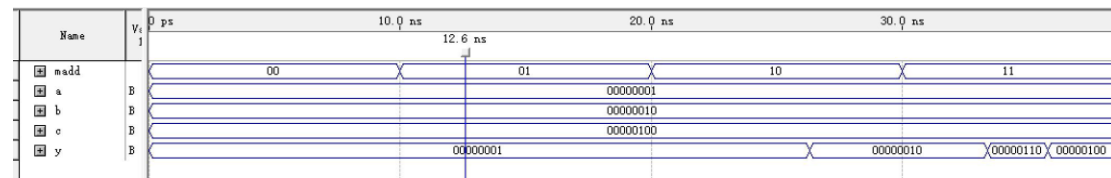
Madd=11 时，控制输出高阻态，正确

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

步骤：新建一个 vector waveform file。通过操作：右击 - 【insert】 - 【insert node or bus】 - 【node finder】 (pins=all; 【list】) - 【>>】 - 【ok】 - 【ok】。接着设置输入波形，然后点击【start simulation】开始仿真，查看时序仿真输出波形图。

时序仿真图：



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】

Compilation Report - Timing Analyzer Summary										
Timing Analyzer Summary										
Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths		
1 Worst-case tpd	N/A	None	19.100 ns	b[7]	y[7]	--	--	0		
2 Total number of failed paths								0		

Timing Analyzer Summmary 图

tpd					
	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	19.100 ns	b[7]	y[7]
2	N/A	None	18.700 ns	a[7]	y[7]
3	N/A	None	18.300 ns	b[6]	y[6]
4	N/A	None	18.300 ns	a[5]	y[5]
5	N/A	None	18.300 ns	b[4]	y[4]
6	N/A	None	18.300 ns	a[4]	y[4]
7	N/A	None	18.200 ns	b[3]	y[3]
8	N/A	None	18.200 ns	a[3]	y[3]
9	N/A	None	18.200 ns	b[2]	y[2]
10	N/A	None	18.200 ns	a[2]	y[2]
11	N/A	None	18.100 ns	a[1]	y[1]
12	N/A	None	17.900 ns	madd[1]	y[7]
13	N/A	None	17.900 ns	madd[1]	y[6]
14	N/A	None	17.900 ns	a[6]	y[6]
15	N/A	None	17.900 ns	madd[1]	y[5]
16	N/A	None	17.900 ns	b[5]	y[5]
17	N/A	None	17.900 ns	madd[1]	y[4]
18	N/A	None	17.900 ns	madd[1]	y[3]
19	N/A	None	17.900 ns	madd[1]	y[2]
20	N/A	None	17.900 ns	madd[1]	y[1]
21	N/A	None	17.900 ns	madd[1]	y[0]

tpd					
	Slack	Required P2P Time	Actual P2P Time	From	To
21	N/A	None	17.900 ns	madd[1]	y[0]
22	N/A	None	17.400 ns	madd[0]	y[7]
23	N/A	None	17.400 ns	madd[0]	y[6]
24	N/A	None	17.400 ns	madd[0]	y[5]
25	N/A	None	17.400 ns	madd[0]	y[4]
26	N/A	None	17.400 ns	madd[0]	y[3]
27	N/A	None	17.400 ns	madd[0]	y[2]
28	N/A	None	17.400 ns	madd[0]	y[1]
29	N/A	None	17.400 ns	madd[0]	y[0]
30	N/A	None	16.300 ns	b[1]	y[1]
31	N/A	None	16.300 ns	b[0]	y[0]
32	N/A	None	16.300 ns	a[0]	y[0]
33	N/A	None	15.700 ns	c[7]	y[7]
34	N/A	None	14.900 ns	c[6]	y[6]
35	N/A	None	14.900 ns	c[5]	y[5]
36	N/A	None	14.800 ns	c[3]	y[3]
37	N/A	None	14.800 ns	c[2]	y[2]
38	N/A	None	14.700 ns	c[1]	y[1]
39	N/A	None	14.500 ns	c[4]	y[4]
40	N/A	None	12.900 ns	c[0]	y[0]

Tpd 图

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 19.100ns。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 a[7]到 y[7] 的 tpd 为 18.700ns。

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。

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:
D:/qua/quartus/

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: <None>

Timing analysis: <None>

Operating conditions:

Core voltage: 5.0V

Junction temperature range: 0-85 °C

< Back Next > Finish 取消

B) 编写源代码

```

1  module shift(fbus,flbus,frbus,a,w,cf);
2      input wire fbus,flbus,frbus;
3      input wire[7:0] a;
4      output reg[7:0] w;
5      output reg cf;
6
7      always @(*)
8      begin
9          cf=0;
10         if(fbus==1'b1&&flbus==1'b0&&frbus==1'b0) w=a;
11         else if(fbus==1'b0&&flbus==1'b1&&frbus==1'b0)
12         begin
13             w[7:1]=a[6:0];
14             w[0]=a[7];
15             cf=a[7];
16         end
17
18         else if(fbus==1'b0&&flbus==1'b0&&frbus==1'b1)
19         begin
20             w[6:0]=a[7:1];
21             w[7]=a[0];
22             cf=a[0];
23         end
24
25         else if(fbus==1'b0&&flbus==1'b0&&frbus==1'b0)
26         begin
27             w=8'hZZ;
28             cf=0;
29         end
30
31         else w=8'hZZ;
32     end
33
34 endmodule

```

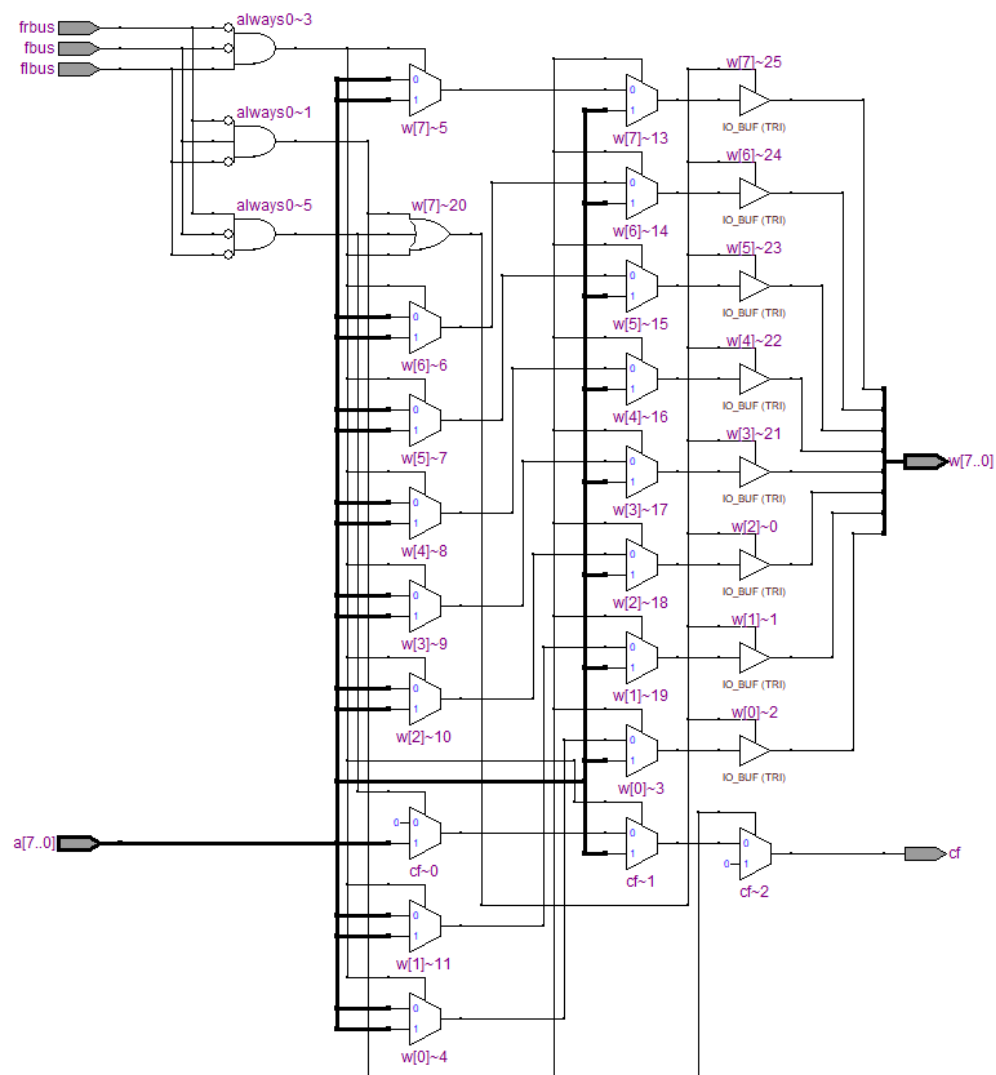
C) 编译与调试 (包含编译调试过程中的错误、警告信息以及资源消耗)

Type	Message
	Info: Fitter placement operations ending: elapsed time is 00:00:00
	Info: Fitter routing operations beginning
	Info: Fitter routing operations ending: elapsed time is 00:00:00
	Info: Quartus II Fitter was successful. 0 errors, 1 warning
	Info: *****
	Info: Running Quartus II Assembler
	Info: Command: quartus_asm --read_settings_files=off --write_settings_files=off shift -c shift
	Info: Assembler is generating device programming files
System (2) Processing (36) Extra Info Info (35) Warning (1) Critical Warning Error Suppressed Flag	
Message: 0 of 75	
Location:	

资源消耗

Flow Status	Successful - Tue Nov 29 17:19:18 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web
Revision Name	shift
Top-level Entity Name	shift
Family	FLEX10K
Device	EPF10K10TI144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	21 / 576 (4 %)
Total pins	20 / 102 (20 %)
Total memory bits	0 / 6,144 (0 %)

D) RTL 视图

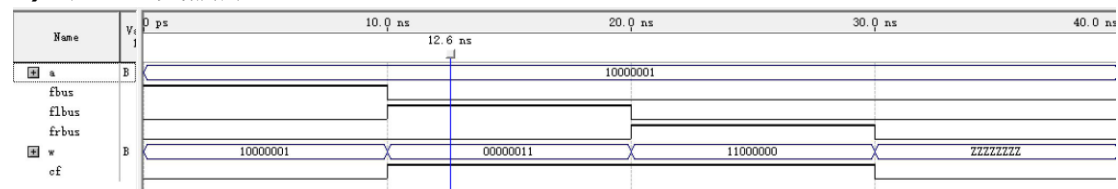


结果分析及结论：

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

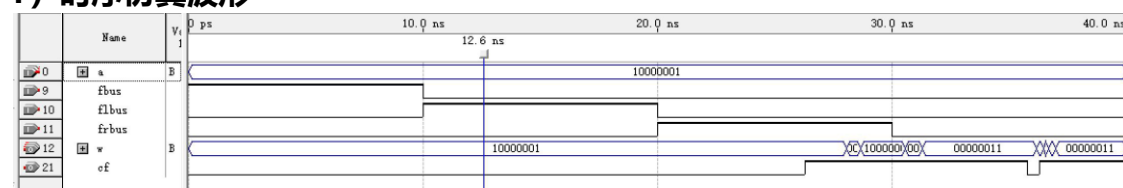
当 fbus=0, frbus=1, flbus=0, 执行右移，输出等于输入右移移位，有进位的话 cf 为 1

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	21.300 ns	fbus	w[0]	--	--	0	
2 Total number of failed paths:								0	

tpd					
	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	21.300 ns	fbus	w[7]
2	N/A	None	21.300 ns	fbus	w[7]
3	N/A	None	21.300 ns	fbus	w[2]
4	N/A	None	21.300 ns	fbus	w[2]
5	N/A	None	21.300 ns	fbus	w[1]
6	N/A	None	21.300 ns	fbus	w[1]
7	N/A	None	21.300 ns	fbus	w[0]
8	N/A	None	21.300 ns	fbus	w[0]
9	N/A	None	20.800 ns	fbus	w[7]
10	N/A	None	20.800 ns	fbus	w[2]
11	N/A	None	20.800 ns	fbus	w[1]
12	N/A	None	20.800 ns	fbus	w[0]
13	N/A	None	20.600 ns	fbus	w[6]
14	N/A	None	20.600 ns	fbus	w[6]
15	N/A	None	20.600 ns	fbus	w[5]
16	N/A	None	20.600 ns	fbus	w[5]
17	N/A	None	20.600 ns	fbus	w[4]
18	N/A	None	20.600 ns	fbus	w[4]
19	N/A	None	20.600 ns	fbus	w[3]
20	N/A	None	20.600 ns	fbus	w[3]
21	N/A	None	20.100 ns	fbus	w[6]
22	N/A	None	20.100 ns	fbus	w[5]
23	N/A	None	20.100 ns	fbus	w[4]
24	N/A	None	20.100 ns	fbus	w[3]
25	N/A	None	19.300 ns	a[6]	w[7]
26	N/A	None	19.300 ns	a[3]	w[2]
27	N/A	None	19.000 ns	a[1]	w[2]

tpd					
	Slack	Required P2P Time	Actual P2P Time	From	To
33	N/A	None	18.300 ns	a[5]	w[4]
34	N/A	None	18.300 ns	a[4]	w[3]
35	N/A	None	17.500 ns	a[7]	w[0]
36	N/A	None	17.400 ns	a[2]	w[1]
37	N/A	None	17.300 ns	a[0]	w[7]
38	N/A	None	17.300 ns	a[0]	w[1]
39	N/A	None	16.900 ns	a[7]	w[6]
40	N/A	None	16.700 ns	a[2]	w[3]
41	N/A	None	16.400 ns	a[0]	cf
42	N/A	None	16.300 ns	fbus	cf
43	N/A	None	16.300 ns	fbus	cf
44	N/A	None	16.300 ns	fbus	cf
45	N/A	None	16.100 ns	a[1]	w[1]
46	N/A	None	15.700 ns	a[6]	w[6]
47	N/A	None	15.700 ns	a[3]	w[3]
47	N/A	None	15.700 ns	a[3]	w[3]
48	N/A	None	15.400 ns	a[5]	w[5]
49	N/A	None	15.400 ns	a[4]	w[4]
50	N/A	None	14.600 ns	a[7]	w[7]
51	N/A	None	14.500 ns	a[2]	w[2]
52	N/A	None	14.400 ns	a[0]	w[0]
53	N/A	None	13.100 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 21.300ns。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 21.300ns。

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。

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)

New Project Wizard: Summary [page 5 of 5] ✕

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

Project directory:
D:/qua/quarter/shift/

Project name: controller

Top-level design entity: controller

Number of files added: 0

Number of user libraries added: 0

Device assignments:

Family name: FLEX10K

Device: EPF10K10T1144-4

EDA tools:

Design entry/synthesis: <None>

Simulation: <None>

Timing analysis: <None>

Operating conditions:

Core voltage: 5.0V

Junction temperature range: 0-85℃

< Back

Next >

Finish



取消

B) 编写源代码

```
con_signal.v
1 module con_signal(
2     input mova,movb,move,add,sub,andl,notl,rsr,rsl,jmp,jz,
3         jc,inl,outl,nop,halt,sm,c,z,
4     input [7:0] ir,
5     output reg[3:0] alu_s,
6     output reg[1:0] madd,reg_ra,reg_wa,
7     output reg pc_ld,pc_inc,reg_we,ram_xl,ram_dl,alu_m,shi_fbus,
8         shi_fibus,shi_frbus,ir_ld,cf_en,zf_en,sm_en,in_en,out_en);
9
10    always @(*) begin
11        sm_en=~halt;
12        ir_ld=~sm;
13        ram_dl=(~sm)|movc|jmp|(jz&z)|(jc&c);
14        ram_xl=movb;
15        shi_fbus=mova|movb|add|sub|andl|notl|outl;
16        shi_fibus=rsl;
17        shi_frbus=rsr;
18        alu_s=ir[7:4];
19        alu_m=andl|notl|add|sub|rsr|rsl|outl;
20        cf_en=add|sub|rsr|rsl;
21        zf_en=add|sub;
22        pc_ld=jmp|(jc&c)|(jz&z);
23        pc_inc=(~sm)|(jc&(~c))|(jz&(~z));
24        reg_we=~(sm&(mova|movc|add|sub|andl|notl|rsr|rsl|inl));
25        reg_ra=ir[1:0];
26        reg_wa=ir[3:2];
27        in_en=inl;
28        out_en=outl;
29        if(sm==1&&movc==1) madd=2'b01;
30        else if(sm==1&&movb==1) madd=2'b10;
31        else madd=2'b00;
32    end
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)

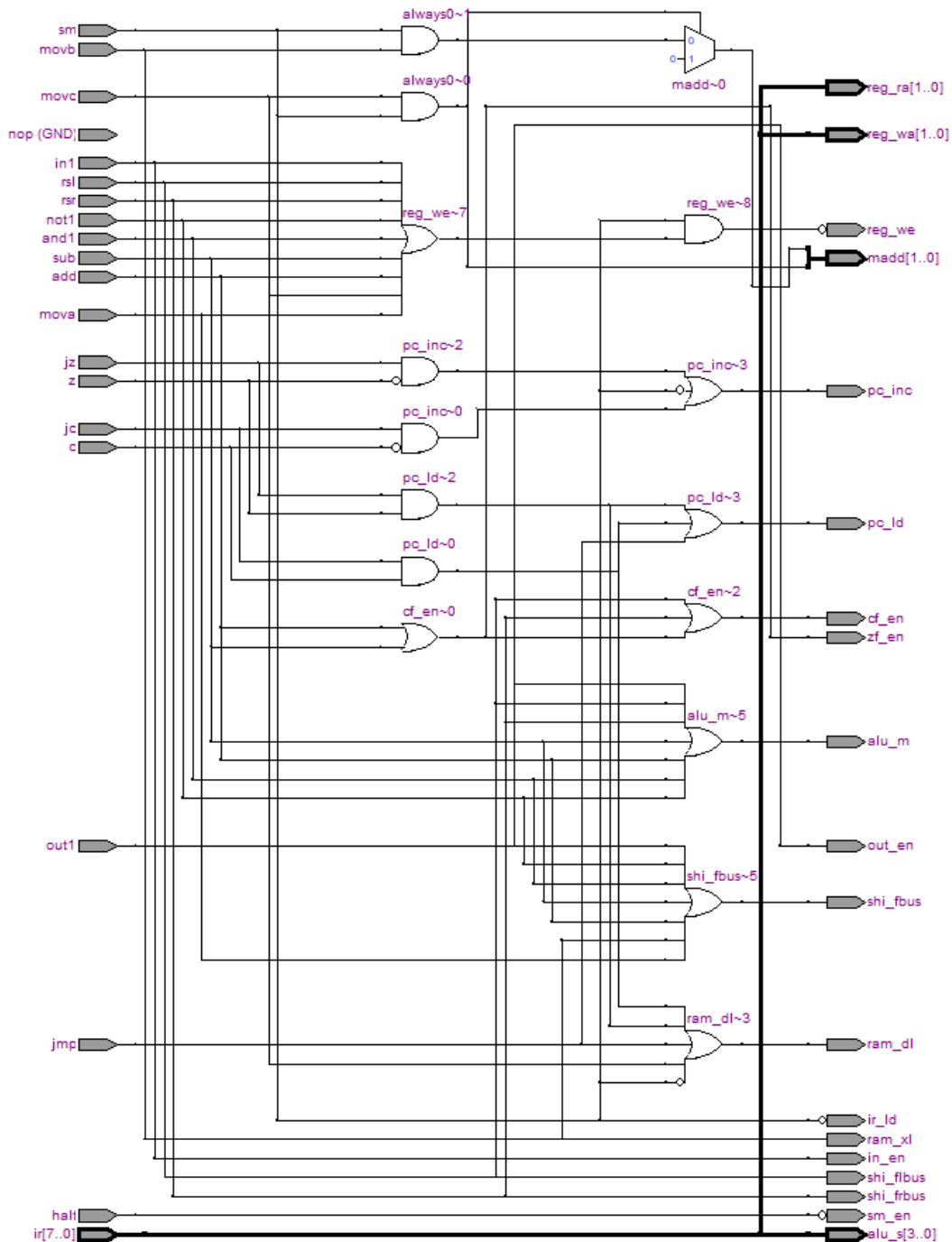
警告信息

type	Message
	Warning: Design contains 1 input pin(s) that do not drive logic
	Warning: Feature SignalProbe is not available with your current license

资源占用

Flow Status	Successful - Tue Nov 29 17:23:55 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	con_signal
Top-level Entity Name	con_signal
Family	FLEX10K
Device	EPF10K10TI144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	30 / 576 (5 %)
Total pins	52 / 102 (51 %)
Total memory bits	0 / 6,144 (0 %)

D) RTL 视图

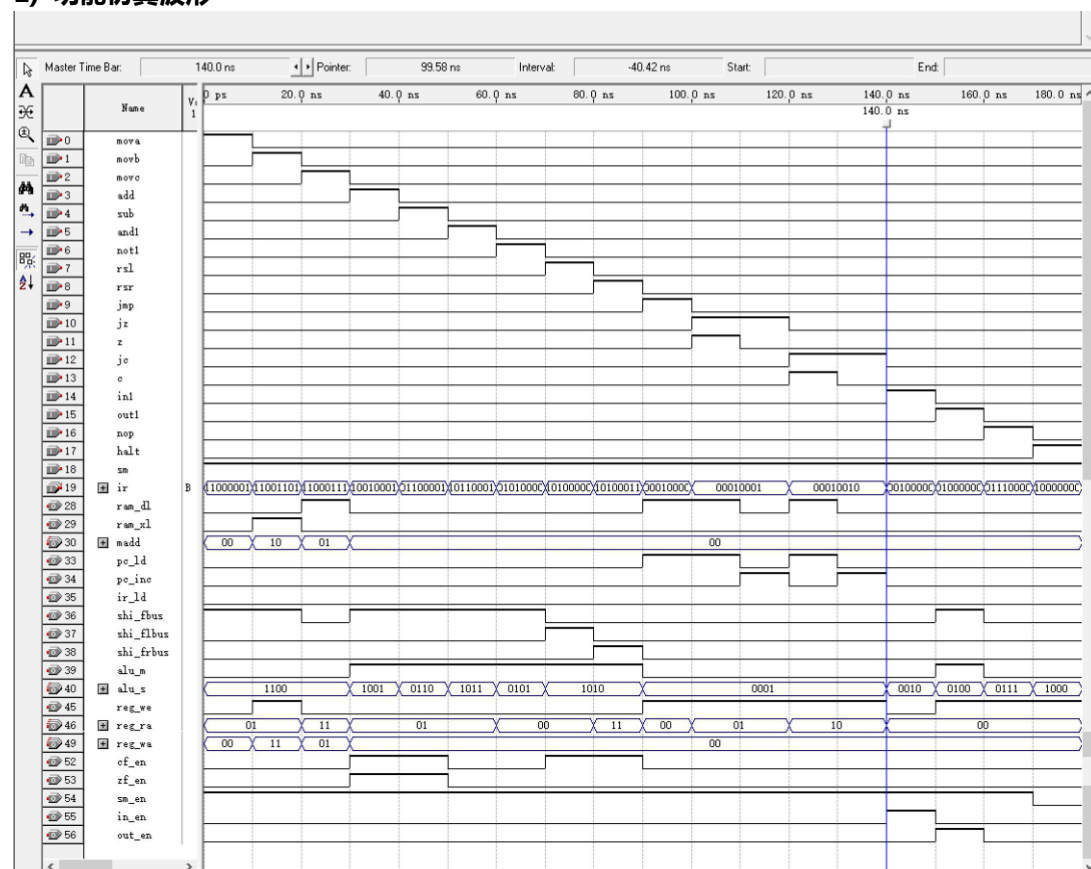


结果分析及结论：

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 1010，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) in1 指令执行时, sm_en 和 in_en 输出 1, 其他输出 0, 正确

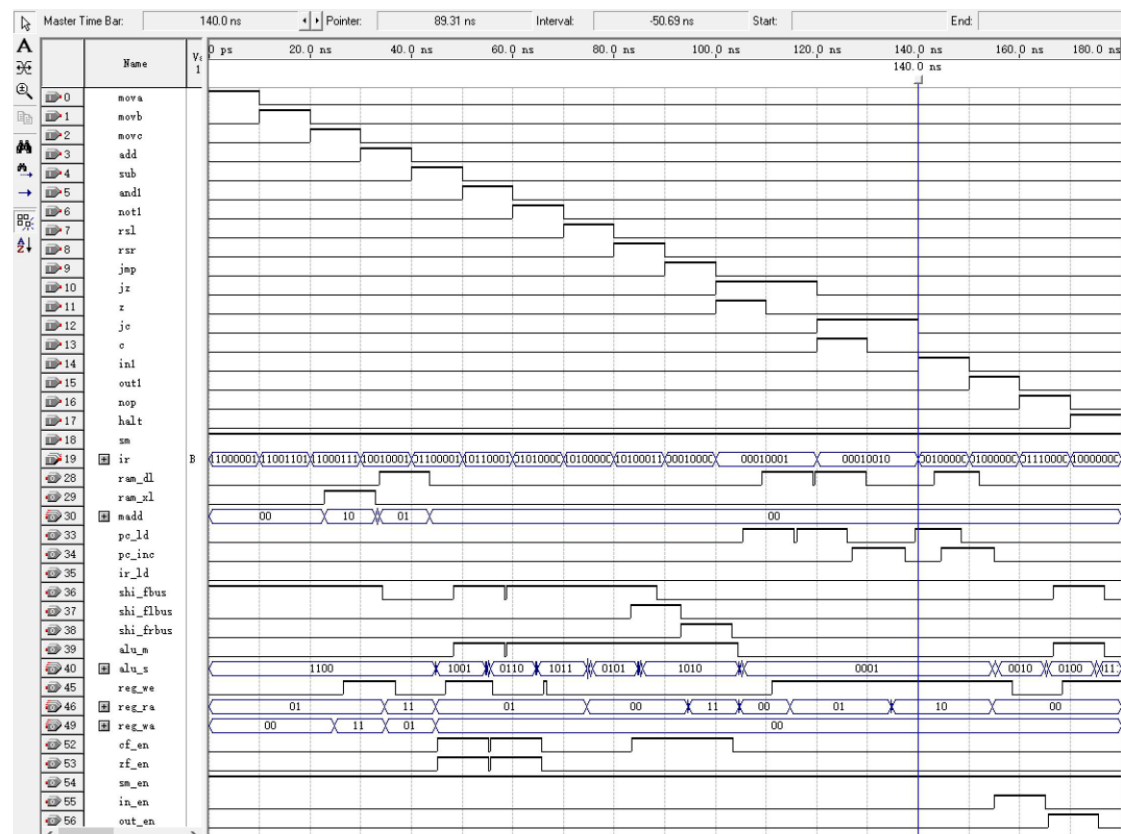
(13) out1 指令执行时, sm_en 和 out_en 输出 1, 其他输出 0, 正确

(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。

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

Compilation Report - Timing Analyzer Summary										
Timing Analyzer Summary										
Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths		
1 Worst-case tpd	N/A	None	26.600 ns	sub	reg_we	--	--	0		
2 Total number of failed paths								0		

tpd						tpd					
	Slack	Required P2P Time	Actual P2P Time	From	To		Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	26.600 ns	sub	reg_we	34	N/A	None	15.600 ns	out1	out_en
2	N/A	None	26.300 ns	and1	reg_we	35	N/A	None	15.500 ns	jmp	pc_ld
3	N/A	None	26.300 ns	not1	reg_we	36	N/A	None	15.200 ns	add	zf_en
4	N/A	None	26.100 ns	add	reg_we	37	N/A	None	15.200 ns	add	cf_en
5	N/A	None	23.100 ns	jc	ram_dl	38	N/A	None	15.100 ns	in1	in_en
6	N/A	None	22.100 ns	c	ram_dl	39	N/A	None	15.000 ns	jc	pc_inc
7	N/A	None	21.700 ns	rsl	reg_we	40	N/A	None	14.900 ns	halt	sm_en
8	N/A	None	21.200 ns	rsr	reg_we	41	N/A	None	14.900 ns	ir[6]	alu_s[2]
9	N/A	None	19.700 ns	jz	ram_dl	42	N/A	None	14.800 ns	ir[3]	reg_wa[1]
10	N/A	None	19.700 ns	z	ram_dl	43	N/A	None	14.800 ns	ir[2]	reg_wa[0]
11	N/A	None	19.400 ns	jc	pc_ld	44	N/A	None	14.700 ns	ir[1]	reg_ra[1]
12	N/A	None	19.200 ns	jmp	ram_dl	45	N/A	None	14.700 ns	ir[4]	alu_s[0]
13	N/A	None	18.800 ns	sub	shi_fbus	46	N/A	None	14.600 ns	sm	reg_we
14	N/A	None	18.800 ns	sub	alu_m	47	N/A	None	14.600 ns	ir[0]	reg_ra[0]
15	N/A	None	18.500 ns	and1	shi_fbus	48	N/A	None	14.600 ns	ir[7]	alu_s[3]
16	N/A	None	18.500 ns	not1	shi_fbus	49	N/A	None	14.500 ns	c	pc_inc
17	N/A	None	18.500 ns	and1	alu_m	50	N/A	None	14.400 ns	rsl	alu_m
15	N/A	None	18.500 ns	and1	shi_fbus	51	N/A	None	14.400 ns	rsr	alu_m
16	N/A	None	18.500 ns	not1	shi_fbus	52	N/A	None	14.200 ns	movb	shi_fbus
17	N/A	None	18.500 ns	and1	alu_m	53	N/A	None	14.200 ns	movb	shi_fbus
18	N/A	None	18.500 ns	not1	alu_m	54	N/A	None	13.800 ns	sm	ram_dl
19	N/A	None	18.500 ns	in1	reg_we	55	N/A	None	13.600 ns	movc	ram_dl
20	N/A	None	18.400 ns	c	pc_ld	56	N/A	None	13.600 ns	movc	madd[0]
21	N/A	None	18.300 ns	add	shi_fbus	57	N/A	None	13.500 ns	rsl	cf_en
22	N/A	None	18.300 ns	add	alu_m	58	N/A	None	13.500 ns	rsr	cf_en
23	N/A	None	17.400 ns	jz	pc_inc	59	N/A	None	13.500 ns	sm	ir_ld
24	N/A	None	16.900 ns	z	pc_inc	60	N/A	None	13.400 ns	sm	madd[0]
25	N/A	None	16.800 ns	movc	reg_we	61	N/A	None	13.200 ns	rsr	shi_fbus
26	N/A	None	16.700 ns	out1	shi_fbus	62	N/A	None	13.200 ns	rsl	shi_fbus
27	N/A	None	16.700 ns	out1	alu_m	63	N/A	None	13.100 ns	movc	madd[1]
28	N/A	None	16.600 ns	movb	reg_we	64	N/A	None	12.900 ns	movb	ram_xl
29	N/A	None	16.000 ns	jz	pc_ld	65	N/A	None	12.900 ns	movb	madd[1]
30	N/A	None	16.000 ns	z	pc_ld	66	N/A	None	12.800 ns	sm	madd[1]
28	N/A	None	16.600 ns	movb	reg_we	67	N/A	None	12.200 ns	sm	pc_inc
29	N/A	None	16.000 ns	jz	pc_ld						
30	N/A	None	16.000 ns	z	pc_ld						

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fibus和fbus为1时，cf=a[7]，当frbus和fbus为1时，cf=a[0]，当只有fbus为1时，cf=0

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、心得体会及建议

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。

答：这次实验我掌握了选择器，移位逻辑，控制器的工作原理，对于模型机更加了解，在过程中对于控制信号逻辑的知识有点模糊，通过请教同学上网查询了解了工作情况

2、对本实验内容、过程和方法的改进建议（可选项）。

无