实验三 模型机组合部件的实现（二）

班级 计科 2003 姓名 袁鹏 学号 202008010321

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。

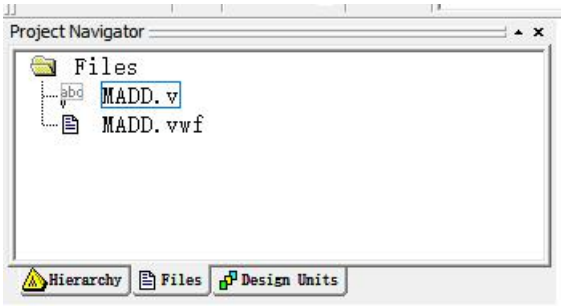
二、实验内容

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。

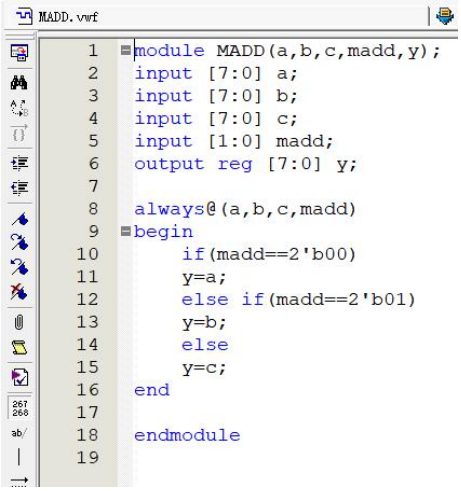
三、实验过程

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）



B）编写源代码

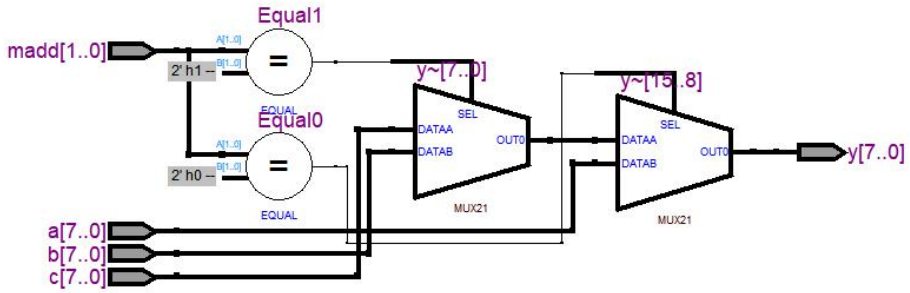


C）编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）

Flow Status	Successful - Sat Dec 11 10:40:02 2021
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	MADD
Top-level Entity Name	MADD
Family	Stratix II
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	8 / 12,480 (< 1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	34 / 343 (10 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Type	Message
⚠	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: Some pins have incomplete I/O assignments. Refer to the I/O Assignment Warnings report for details
⚠	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'.

D）RTL 视图

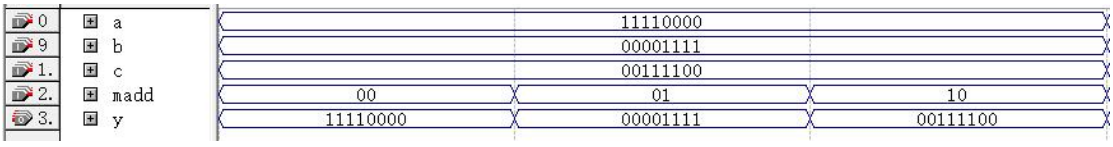


视图分析及结论：

RTL 视图中有两个判等符号而两个八重 2-1 多路复用器。

使用两个八重的 2-1 多路复用器从输入 a，b，c 中选择一个输出。实现三选一的功能，虽然没有出现 madd=2'b11 的情况，但是为了不出现锁存器，将 11 的情况并入到 10 中。由于 11 的情况根本不会出现，所以将其并入 10 的情况中不会对 CPU 产生影响。

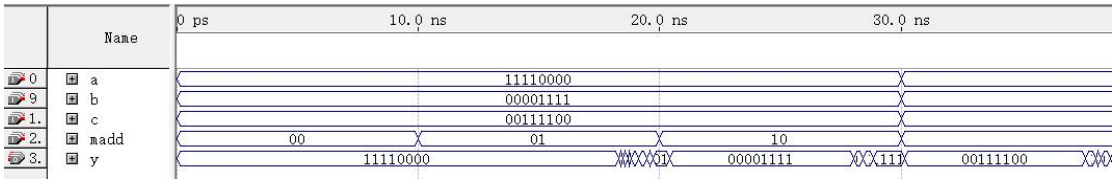
E）功能仿真波形



结果分析及结论：

madd 输入 00 的时候，3-1 多路复用器选择指令计数器通道传来的信号。当 madd 输入 01 的时候，3-1 多路复用器选择源寄存器通道传来的信号，此时 ram 作为数据源，从 ram 中读取数据送到总线。当 madd 输入 10 的时候，3-1 多路复用器选择目的寄存器通道传来的信号，此时 ram 作为目的，ram 对应地址写入总线中的数据。

F）时序仿真波形



结果分析及结论：观察波形可知，第一次正常的输出在 17ns 消失，第二次正常的输出在 20ns 左右出现，在 27ns 左右消失，第三次正常的输出在 30ns 左右出现。

可以得出结论：因竞争冒险会导致不正常输出，波形中出现噪声，两个正常输出之间的噪声约为 3ns，而每个输出的延时约为 10ns

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	10.407 ns	madd[0]	y[0]	--	--	0
2	Total number of failed paths								0

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	10.407 ns	madd[0]	y[0]
2	N/A	None	10.393 ns	b[0]	y[0]
3	N/A	None	10.133 ns	madd[1]	y[0]
4	N/A	None	9.882 ns	a[0]	y[0]
5	N/A	None	9.729 ns	madd[0]	y[7]
6	N/A	None	9.480 ns	madd[0]	y[3]
7	N/A	None	9.454 ns	madd[1]	y[7]
8	N/A	None	9.386 ns	c[0]	y[0]
9	N/A	None	9.205 ns	madd[1]	y[3]
10	N/A	None	9.167 ns	c[7]	y[7]
11	N/A	None	9.118 ns	madd[0]	y[5]
12	N/A	None	9.111 ns	b[5]	y[5]
13	N/A	None	9.080 ns	b[7]	y[7]
14	N/A	None	8.895 ns	a[7]	y[7]
15	N/A	None	8.846 ns	madd[1]	y[5]
16	N/A	None	8.812 ns	madd[0]	y[6]
17	N/A	None	8.810 ns	b[3]	y[3]
18	N/A	None	8.691 ns	madd[0]	y[4]
19	N/A	None	8.676 ns	c[3]	y[3]
20	N/A	None	8.609 ns	a[3]	y[3]
21	N/A	None	8.537 ns	madd[1]	y[6]
22	N/A	None	8.480 ns	a[6]	y[6]
23	N/A	None	8.473 ns	madd[0]	y[2]
24	N/A	None	8.419 ns	madd[1]	y[4]
25	N/A	None	8.297 ns	madd[0]	y[1]
26	N/A	None	8.267 ns	c[5]	y[5]
27	N/A	None	8.198 ns	madd[1]	y[2]
28	N/A	None	8.075 ns	a[5]	y[5]
29	N/A	None	8.066 ns	c[4]	y[4]
30	N/A	None	8.064 ns	b[4]	y[4]
31	N/A	None	8.038 ns	c[6]	y[6]
32	N/A	None	8.031 ns	b[6]	y[6]
33	N/A	None	8.023 ns	madd[1]	y[1]
34	N/A	None	7.850 ns	a[4]	y[4]
35	N/A	None	7.706 ns	b[2]	y[2]
36	N/A	None	7.678 ns	b[1]	y[1]
37	N/A	None	7.639 ns	c[2]	y[2]
38	N/A	None	7.609 ns	a[2]	y[2]
39	N/A	None	7.242 ns	a[1]	y[1]
40	N/A	None	7.064 ns	c[1]	y[1]

结果分析及结论：

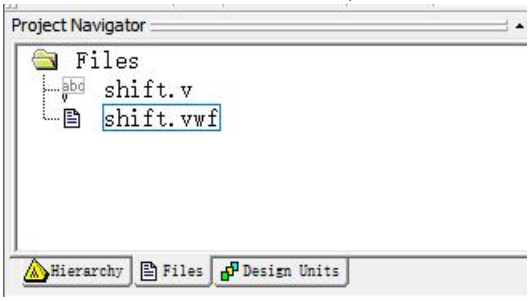
最长延时为 10.407ns，最短的延时为 7.064ns。

由于不同路径的延时不同，所以两条不同路径的信号到达输出端的时间不同步，这会导致竞争冒险现象。最差情况下的竞争冒险现象会持续约 3ns，即输出既包括最长延时路径传来的

信号，又包括最短延时路径传来的信号时，会产生约 3ns 的毛刺。这与时序仿真的波形图一致。

2、移位逻辑

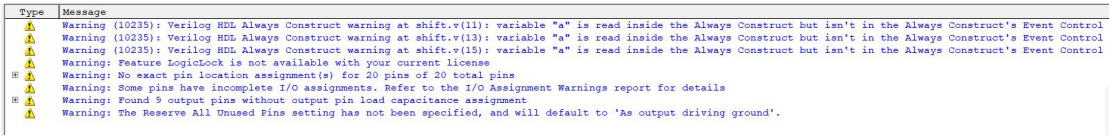
A）创建工程（选择的芯片为 family=FLEX10K；name=EPF10K20TI144-4）



B）编写源代码

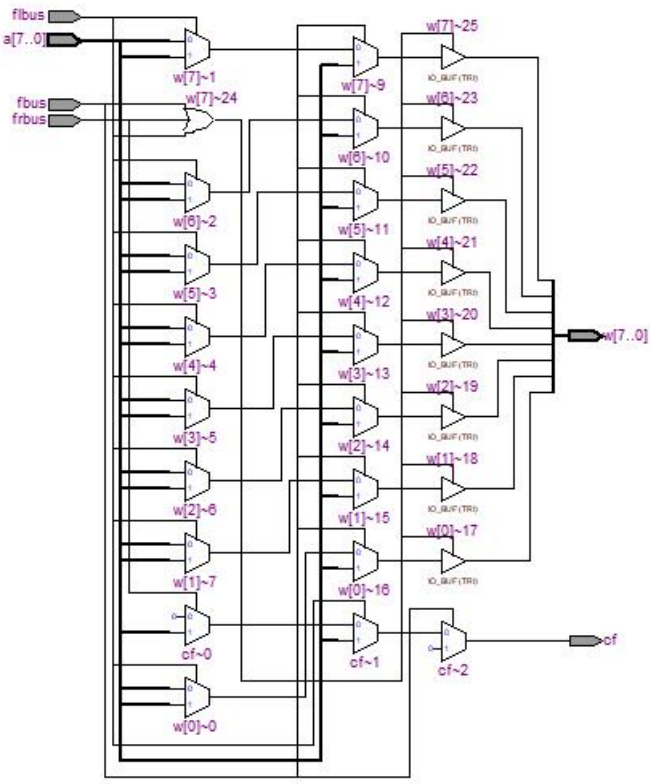
```
1  module shift(fbus,flbus,frbus,a,w,cf);
2   input fbus,flbus,frbus;
3   input [7:0]a;
4   output reg[7:0]w;
5   output reg cf;
6
7   always@(fbus,flbus,frbus,w,cf)
8  begin
9       cf=0;
10      if(fbus==1)
11      w=a;
12      else if(flbus==1)
13      begin {cf,w}=a<<1; w[0]=cf; end
14      else if(frbus==1)
15      begin cf=a[0]; w=a>>1; w[7]=cf; end
16      else
17      w=8'hZZ;
18  end
19  endmodule
20
```

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）



Flow Status	Successful - Sat Dec 11 11:11:08 2021
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	shift
Top-level Entity Name	shift
Family	Stratix II
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	10 / 12,480 (< 1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	20 / 343 (6 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

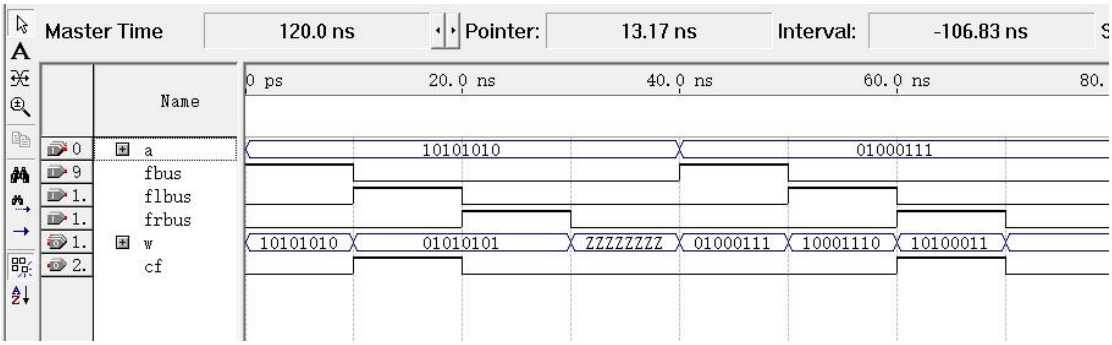
D) RTL 视图



结果分析及结论：

移位逻辑主要由 2-1 多路复用器和缓冲器构成。通过 2-1 多路复用器实现循环移位操作并将移出去的 1 送入 cf 中，通过缓冲器与总线直接相连，既可以让数据通过，也可以呈高阻态，与总线断开。

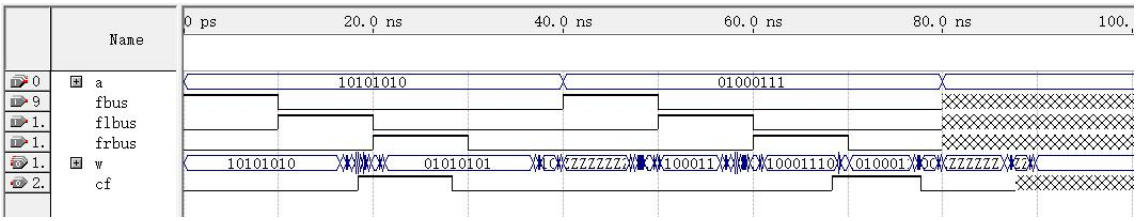
E）功能仿真波形



结果分析及结论：

fbus，flbus，frbus 中最多只能有一个为 1，分别执行不移位，循环左移，循环右移操作，相应操作完成后将数据总到总线中。若三者全为 0，此时输出呈高阻态，与总线断开。当移出去的位为 1 时，cf 输出为 1。

F）时序仿真波形



结果分析及结论：观察波形可知，第一次正常的输出在 17ns 消失，第二次正常的输出在 21ns 左右出现，在 37ns 左右消失，第三次正常的输出在 40ns 左右出现。

可以得出结论：因竞争冒险会导致不正常输出，波形中出现噪声，两个正常输出之间的噪声约为 3-4ns，而每个输出的延时约为 10ns

G）时序分析

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	11.200 ns	flbus	w[7]
2	N/A	None	10.736 ns	flbus	w[3]
3	N/A	None	10.667 ns	flbus	w[4]
4	N/A	None	10.497 ns	frbus	w[7]
5	N/A	None	10.479 ns	a[2]	w[3]
6	N/A	None	10.447 ns	a[3]	w[4]
7	N/A	None	10.221 ns	a[3]	w[3]
8	N/A	None	10.184 ns	fbus	w[7]
9	N/A	None	10.033 ns	frbus	w[3]
10	N/A	None	9.964 ns	frbus	w[4]
11	N/A	None	9.880 ns	a[0]	w[7]
12	N/A	None	9.766 ns	fbus	w[4]
13	N/A	None	9.720 ns	fbus	w[3]
14	N/A	None	9.547 ns	a[5]	w[4]
15	N/A	None	9.543 ns	a[2]	w[2]
16	N/A	None	9.512 ns	a[7]	w[7]
17	N/A	None	9.356 ns	a[6]	w[7]
18	N/A	None	9.342 ns	a[4]	w[4]
19	N/A	None	9.155 ns	a[3]	w[2]
20	N/A	None	9.126 ns	flbus	w[2]
21	N/A	None	9.111 ns	a[4]	w[3]
22	N/A	None	9.095 ns	flbus	w[5]
23	N/A	None	8.900 ns	flbus	w[0]
24	N/A	None	8.890 ns	flbus	w[6]
25	N/A	None	8.579 ns	a[2]	w[1]
26	N/A	None	8.507 ns	fbus	w[5]
27	N/A	None	8.472 ns	a[1]	w[2]
28	N/A	None	8.343 ns	flbus	cf
29	N/A	None	8.340 ns	frbus	w[2]
30	N/A	None	8.325 ns	a[0]	w[0]
31	N/A	None	8.313 ns	frbus	w[5]
32	N/A	None	8.307 ns	a[6]	w[5]
33	N/A	None	8.257 ns	a[5]	w[5]
34	N/A	None	8.232 ns	fbus	w[6]
35	N/A	None	8.203 ns	a[1]	w[0]
36	N/A	None	8.197 ns	frbus	w[0]
37	N/A	None	8.187 ns	frbus	w[6]
38	N/A	None	8.164 ns	a[7]	w[0]
39	N/A	None	8.162 ns	flbus	w[1]
40	N/A	None	8.122 ns	a[0]	cf
41	N/A	None	8.102 ns	fbus	w[2]
42	N/A	None	8.084 ns	a[4]	w[5]
43	N/A	None	8.081 ns	fbus	w[0]
44	N/A	None	8.037 ns	a[6]	w[6]
45	N/A	None	7.994 ns	a[7]	w[6]
46	N/A	None	7.818 ns	a[5]	w[6]
47	N/A	None	7.674 ns	frbus	cf
48	N/A	None	7.595 ns	a[0]	w[1]
49	N/A	None	7.477 ns	a[1]	w[1]
50	N/A	None	7.456 ns	a[7]	cf
51	N/A	None	7.350 ns	fbus	w[1]
52	N/A	None	7.329 ns	fbus	cf
53	N/A	None	7.318 ns	frbus	w[1]

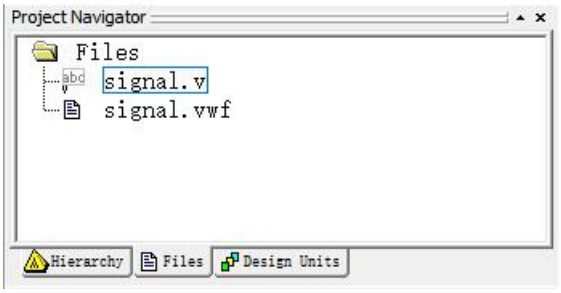
结果分析及结论：

最长延时为 11.2ns，最短的延时为 7.318ns。

由于不同路径的延时不同，所以两条不同路径的信号到达输出端的时间不同步，这会导致竞争冒险现象。最差情况下的竞争冒险现象会持续约 4ns，即输出既包括最长延时路径传来的信号，又包括最短延时路径传来的信号时，会产生约 4ns 的毛刺。这与时序仿真的波形图一致。

3、控制信号产生逻辑

A）创建工程（选择的芯片为 family=FLEX10K；name=EPF10K20TI144-4）



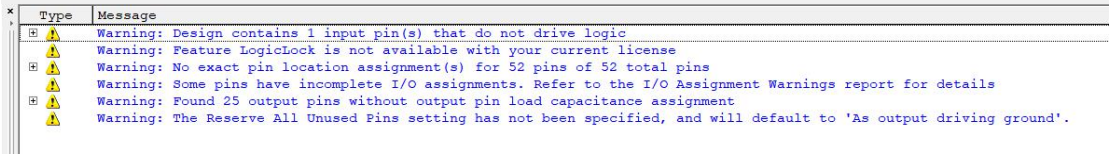
B）编写源代码

```
module signal(mova,movb,movc,add,sub,and1,not1,rsr,rsl,jmp,jz,jc,in1,out1,nop,halt,ir,sm,cf,zf,
reg_ra,reg_wa,reg_we,
madd,
alu_s,alu_m,
shi_fbus,shi_flbus,shi_frbus,
cf_en,zf_en,
pc_ld,pc_in,
ram_xl,ram_dl,
ir_ld,sm_en,
in_en,out_en);
input mova,movb,movc,add,sub,and1,not1,rsr,rsl,jmp,jz,jc,in1,out1,nop,halt,sm,cf,zf;
input [7:0]ir;
output reg [1:0] reg_ra,reg_wa,madd;
output reg [3:0]alu_s;
output reg reg_we,alu_m,
shi_fbus,shi_flbus,shi_frbus,
cf_en,zf_en,
pc_ld,pc_in,
ram_xl,ram_dl,
ir_ld,sm_en,
in_en,out_en;
```

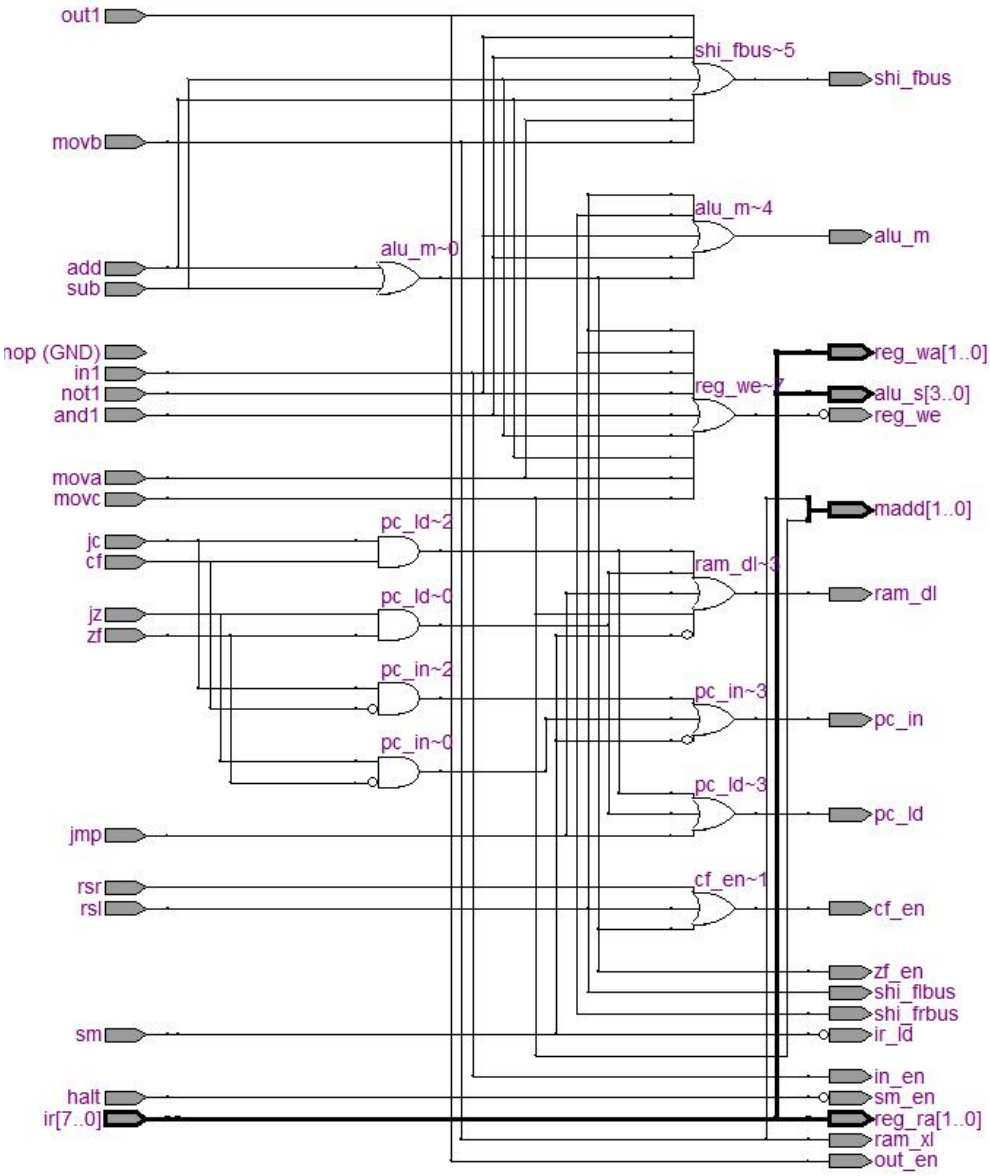
```
always@(mova,movb,movc,add,sub,and1,not1,rsr,rsl,jmp,jz,jc,in1,out1,nop,halt,sm,ir,cf,zf)
begin
alu_s=ir[7:4];
cf_en=add|sub|rsl|rsr;
zf_en=add|sub;
sm_en=~(halt);
in_en=in1;
out_en=out1;
pc_ld=jmp|(jz&zf)|(jc&cf);
pc_in=~(sm)|(jz&(~(zf)))|(jc&(~(cf)));
madd[0]=movc;
madd[1]=movb;
ram_dl=(~(sm))|movc|jmp|(jz&zf)|(jc&cf);
ram_xl=movb;
ir_ld=~(sm);
reg_we=~(mova|movc|add|sub|and1|not1|in1|rsr|rsl);
reg_ra=ir[1:0];
reg_wa=ir[3:2];
alu_m=add|sub|and1|not1|rsr|rsl;
shi_fbus=mova|movb|add|sub|and1|not1|out1;
shi_flbus=rsl;
shi_frbus=rsr;
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）

Flow Status	Successful - Tue Dec 14 12:50:38 2021
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	signal
Top-level Entity Name	signal
Family	Stratix II
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	9 / 12,480 (< 1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	52 / 343 (15 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final



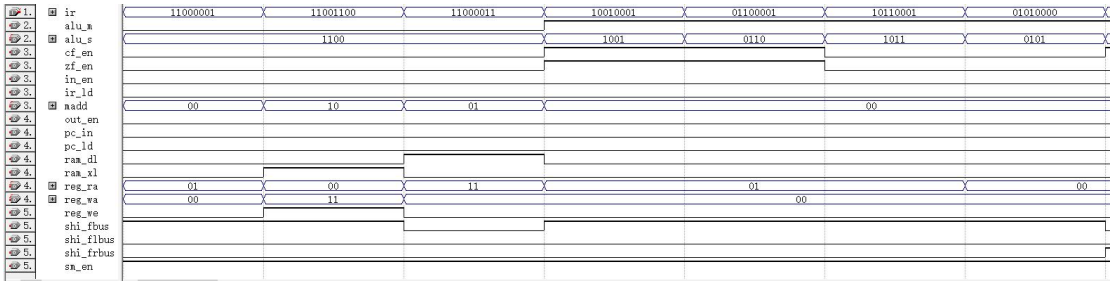
D) RTL 视图



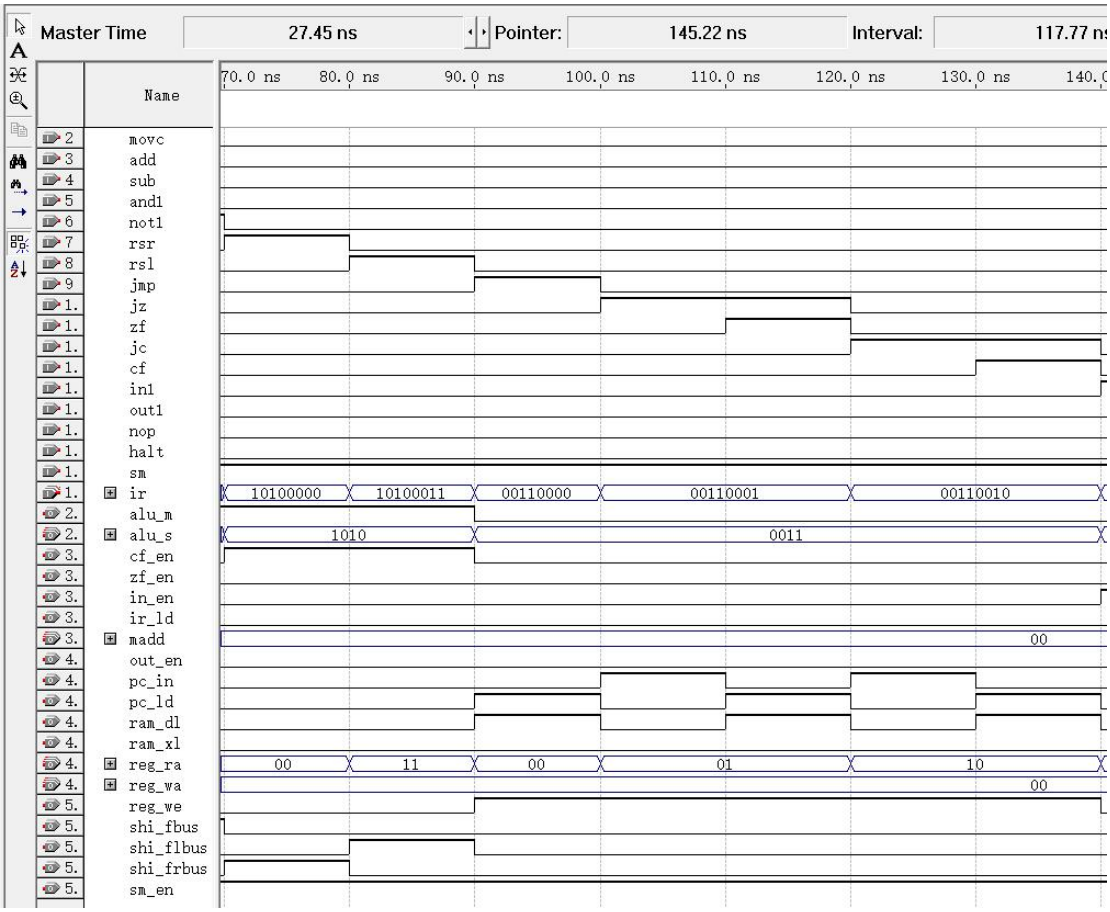
结果分析及结论：

控制信号主要由与门和或门组成。与门用于处理 jc 和 jz 成功或不成功的逻辑关系，或门则是将译码器译码的信号对应的控制信号输出。

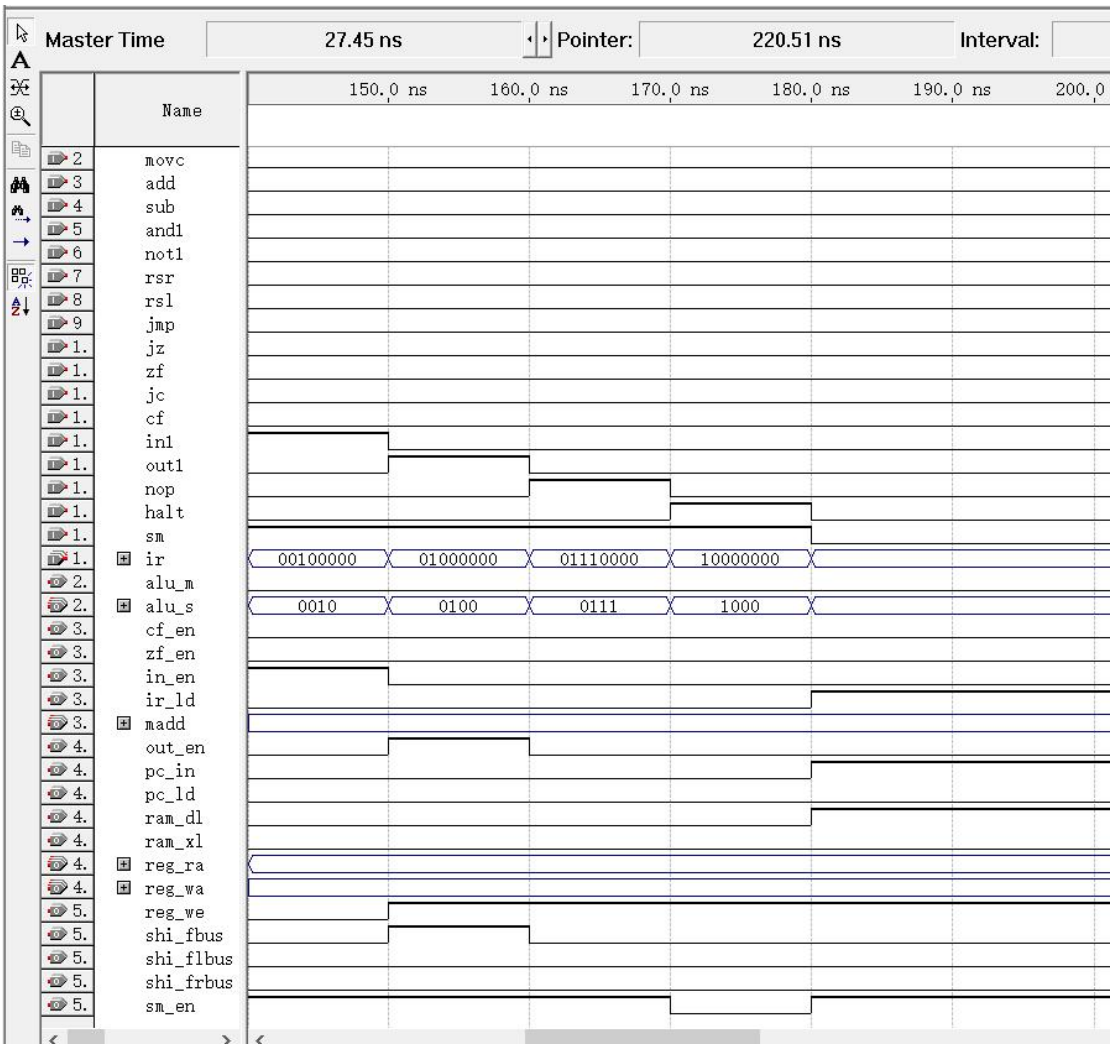
E) 功能仿真波形



三个传送类指令和四个算术逻辑类指令的仿真结果



两个移位逻辑，三个转移类指令：jmp，jz 不成功，jz 成功，jc 不成功，jc 成功的仿真



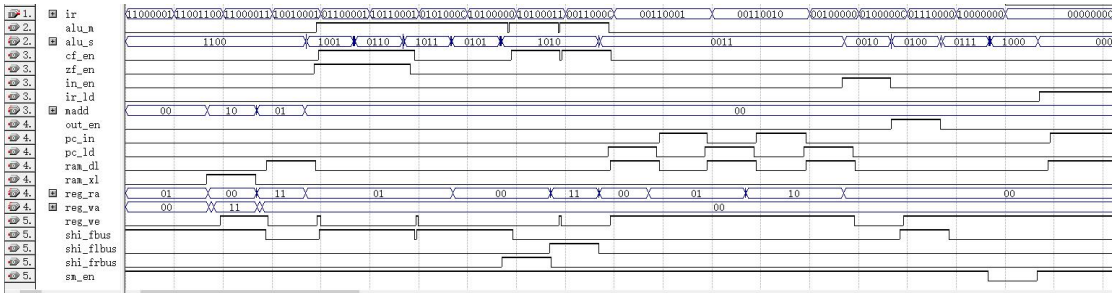
一个输入，一个输出，一个 NOP，一个停机，一个译码的仿真

结果分析及结论：

三个传送类指令涉及到数据的源端和目的端，所以需要控制 madd，ram 的读写，通用寄存器组的写，移位逻辑通道的通和不通。4 个算术逻辑运算类指令和 2 个移位逻辑指令需要从通用寄存器组读和写数据，需要通过数据通路，所以要发出数据通路相关的控制信号。三个转移指令则涉及是否到从 ram 中取指令送到指令计数器中，jmp 是一定会取出指令送到指令计数器，jz 和 jc 则实际是否成功，若不成功则 pc+1，否则与 jmp 一致。out 和 in 指令则与外界输入设备相连，可以通过输入输出设备控制 cpu 的运行，或者查看 cpu 内的数据。执行 nop 指令时不进行任何操作，而执行 halt 停机操作时，由于 sm 不再翻转，cpu 将一直处于执行 halt 操作的状态，cpu 处于停机状态。

功能仿真波形输出的各个值与分析一致，故 cpu 的设计是正确的。

F）时序仿真波形



结果分析及结论：

输出的延时集中在 8-9ns，相比于 madd 和移位逻辑的八位输出，仅为四位输出的 alu_s 因竞争冒险产生的错误输出时间明显短一些。观察到 reg_we 和 fbus 的毛刺现象稍微多一些，而其他的输出则不那么明显。这大概是因为 reg_we 和 fbus 涉及到的输入更多，更容易产生竞争现象，所以毛刺很多。

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	9.940 ns	and1	reg_we	--	--	0
2	Total number of failed paths								0

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	9.940 ns	and1	reg_we
2	N/A	None	9.902 ns	add	reg_we
3	N/A	None	9.667 ns	and1	shi_fbus
4	N/A	None	9.629 ns	add	shi_fbus
5	N/A	None	9.487 ns	sub	reg_we
6	N/A	None	9.472 ns	not1	reg_we
7	N/A	None	9.454 ns	add	cf_en
8	N/A	None	9.427 ns	mova	reg_we
9	N/A	None	9.352 ns	jz	ram_dl
10	N/A	None	9.314 ns	rsl	cf_en
11	N/A	None	9.305 ns	cf	ram_dl
12	N/A	None	9.270 ns	jc	ram_dl
13	N/A	None	9.266 ns	jz	pc_in
14	N/A	None	9.214 ns	sub	shi_fbus
15	N/A	None	9.203 ns	jmp	ram_dl
16	N/A	None	9.199 ns	not1	shi_fbus
17	N/A	None	9.198 ns	sub	cf_en
18	N/A	None	9.188 ns	cf	pc_in
19	N/A	None	9.184 ns	jc	pc_in
20	N/A	None	9.181 ns	movc	reg_we
21	N/A	None	9.176 ns	sm	pc_in
22	N/A	None	9.171 ns	rsl	reg_we
23	N/A	None	9.154 ns	mova	shi_fbus
24	N/A	None	9.150 ns	in1	reg_we
25	N/A	None	9.072 ns	add	alu_m
26	N/A	None	9.017 ns	zf	ram_dl
27	N/A	None	9.015 ns	zf	pc_in
28	N/A	None	8.932 ns	rsl	alu_m
29	N/A	None	8.931 ns	movc	ram_dl
30	N/A	None	8.927 ns	rsr	cf_en

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
31	N/A	None	8.892 ns	jz	pc_ld
32	N/A	None	8.845 ns	cf	pc_ld
33	N/A	None	8.816 ns	sub	alu_m
34	N/A	None	8.810 ns	jc	pc_ld
35	N/A	None	8.778 ns	sm	ram_dl
36	N/A	None	8.743 ns	jmp	pc_ld
37	N/A	None	8.694 ns	movb	shi_fbus
38	N/A	None	8.659 ns	rsr	reg_we
39	N/A	None	8.638 ns	and1	alu_m
40	N/A	None	8.567 ns	add	zf_en
41	N/A	None	8.557 ns	zf	pc_ld
42	N/A	None	8.545 ns	rsr	alu_m
43	N/A	None	8.515 ns	out1	shi_fbus
44	N/A	None	8.311 ns	sub	zf_en
45	N/A	None	8.169 ns	not1	alu_m
46	N/A	None	8.046 ns	ir[2]	reg_wa[0]
47	N/A	None	7.122 ns	ir[4]	alu_s[0]
48	N/A	None	7.077 ns	rsr	shi_frbus
49	N/A	None	7.016 ns	ir[3]	reg_wa[1]
50	N/A	None	6.984 ns	ir[0]	reg_ra[0]
51	N/A	None	6.952 ns	sm	ir_ld
52	N/A	None	6.894 ns	ir[6]	alu_s[2]
53	N/A	None	6.882 ns	ir[1]	reg_ra[1]
54	N/A	None	6.833 ns	rsl	shi_flbus
55	N/A	None	6.771 ns	movb	madd[1]
56	N/A	None	6.750 ns	movc	madd[0]
57	N/A	None	6.674 ns	out1	out_en
58	N/A	None	6.638 ns	ir[7]	alu_s[3]
59	N/A	None	6.619 ns	in1	in_en
60	N/A	None	6.598 ns	movb	ram_xl

61	N/A	None	6.580 ns	ir[5]	alu_s[1]
62	N/A	None	6.527 ns	halt	sm_en

结果分析及结论：
最长延时为 9.94ns，最短的延时为 6.527ns。
由于不同路径的延时不同，所以两条不同路径的信号到达输出端的时间不同步，这会导致竞争冒险现象。最差情况下的竞争冒险现象会持续约 3ns，即输出既包括最长延时路径传来的信号，又包括最短延时路径传来的信号时，会产生约 3ns 的毛刺。但在时序仿真的波形图中未见到有长达 3ns 的错误输出。这主要是因为最多位数的输出为 alu_s，它只有 4 位，四条不同的路径到达输出时，产生的延时基本一直，所以错误的输出没有持续很长时间。
可以推理出一个普遍的结论，当输出只有 1 位时，竞争产生的冒险只有非常短暂的毛刺。当输出的位数不断增加时，由于不同路径到达输出端的延时不同，会导致输出有一段时间都不正常，这段时间的时长为最早的位达到的时间与最晚的位达到的时间之间的差值。

四、思考题

1．移位逻辑不工作时，输出应该为何值？为什么？

当移位逻辑既不执行循环左移或循环右移，又不作为传输通道将数据送至总线时，移位逻辑应当与总线断开，即输出高阻态。否则移位逻辑会将数据送到总线，使得总线当中有多组数据，这会干扰其他模块的正常运转。

2、任选一条指令，介绍指令的过程、信息流动的情况以及执行时控制信号的值。

add 指令：
将数据从源寄存器的输出口和目的寄存器的输出口取出，送到 ALU 中完成加法运算，如果有进位则送入 1 到 cf 中，如果输出结果等于 1 则送入 0 到 zf 中。将输出结果送给移位逻辑后，送到总线中去。在下降沿将总线中的运算结果写入目的寄存器中。
执行 add 操作时，需要从通用寄存器组取数据并且进行算术逻辑运算，需要通过移位逻辑将数据送入总线，需要通用寄存器组可写。因此需要如下控制信号有效：/WE,RA,RWB,M,S[3:0],ZF,CF,FBUS。其中/WE 是低电平有效，其余都是高电平有效。

3．如何产生正确的控制信号以及具体的编程实现？

通过分析每个指令会导致哪些控制信号有效，由此可以得到使得任何一个控制信号有效的所有指令，由于每次只可能执行一个指令，即不会有多个指令的取值为 1，所以对于高电平有效的控制信号只需要将使得该控制信号有效的所有指令算术加（逻辑或也行）起来即可，对于低电平有效的控制信号（如/WE）则还需要取反。

```
alu_s=ir[7:4];
cf_en=add+sub+rsl+rsr;
zf_en=add+sub;
sm_en=~(halt);
in_en=in1;
out_en=out1;
pc_ld=jmp+(jz&zf)+(jc&cf);
pc_in=~(sm)+(jz&(~(zf)))+(jc&(~(cf)));
madd[0]=movc;
madd[1]=movb;
ram_dl=(~(sm))+movc+jmp+(jz&zf)+(jc&cf);
ram_xl=movb;
ir_ld=~(sm);
reg_we=~(mova+movc+add+sub+and1+not1+in1+rsr+rsl);
reg_ra=ir[1:0];
reg_wa=ir[3:2];
alu_m=add+sub+and1+not1+rsr+rsl;
shi_fbus=mova+movb+add+sub+and1+not1+out1;
shi_flbus=rsl;
shi_frbus=rsr;
```

具体的编程实现如图所示

五、实验总结、必得体会及建议

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。

需要掌握的理论：

简易模型机的内部结构和工作原理。

2、8 重 3-1 多路复用器的设计方法。

3、移位逻辑的设计方法。

4、模型机控制信号产生逻辑的设计方法。

遇到的困难：

1、使用 Verilog 编写程序后，仿真发现与 jz 和 jc 有关的波形不正确，但是检查控制信号的分析却并没有发现问题。

2、对位移逻辑的理解存在误区，一开始以为是舍弃最高位（对于逻辑左移而言），后面写完了才发现原来是循环左移和循环右移，移出来的位要放到最低位去。

3、对于执行 halt 停机语句以后，模型机的状况不清楚

解决的办法：

1、Verilog 中&的优先级并没有+的优先级高，一开始我没有打括号，导致仿真波形出错了。使用“（）”小括号将表达式的优先级明确以后，波形就正确了

2、和同学讨论，并且再仔细阅读实验手册后，解决了问题。

3、向老师请教，然后深刻的理解模型机运行的机制，再教其他不懂的同学，就完全明白执行 halt 停机语句以后的情况。

经验教训：

1、对于操作符优先级不明确时，可以打括号来解决问题。

2、实验前要先仔细的阅读实验手册，不能想当然的直接编写代码，可能自己的理解存在误区。通过自习阅读实验手册可以解决大部分问题。

3、多思考，多和同学讨论，多向老师请教不懂的问题，而不是让不懂的问题堆积。