

实验三模型机组合部件的实现（二）

班级人工智能 2103 姓名姚丁钰 学号 202107030125

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

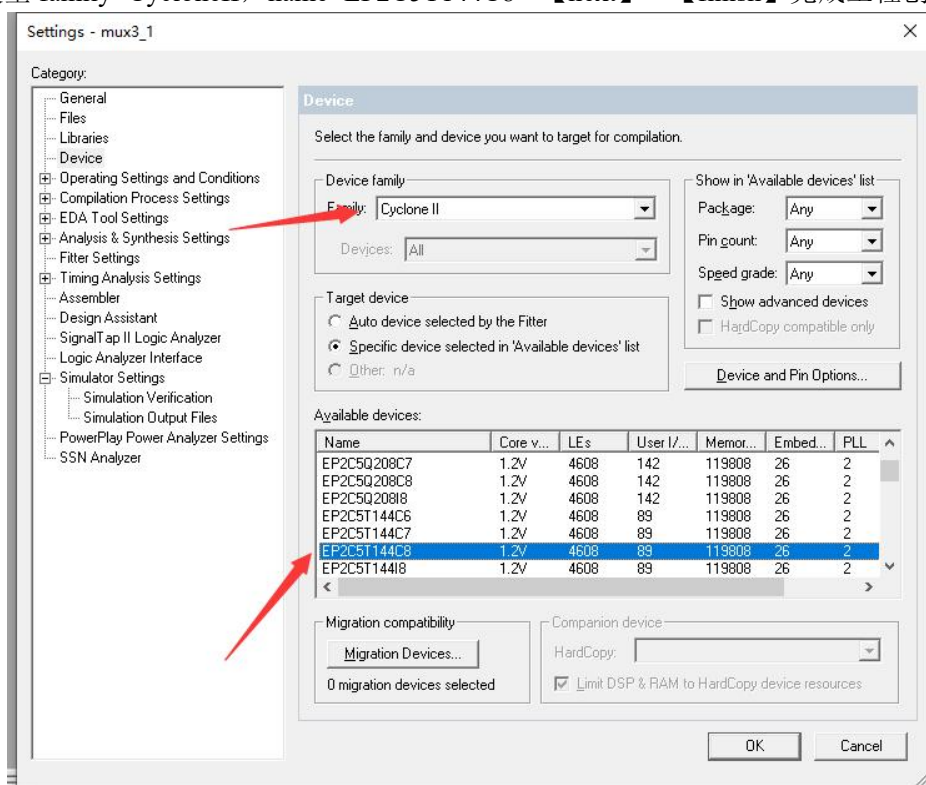
1. 用 VERILOG 语言设计模型机的 8 重 3-1 多路复用器；
2. 用 VERILOG 语言设计模型机的移位模块；
3. 用 VERILOG 语言设计模型机的控制信号产生逻辑。

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=CycloneII; name=EP2C5T144C8）

步骤：【File】->【newprojectwizard】->【next】->【next】->【properties】->【next】->选择芯片类型 family=CycloneII, name=EP2C5T144C8->【next】->【finish】完成工程创建



B) 编写源代码

```

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@(a,b,c,madd)
7   begin
8       if (madd==2'b00)
9           y=a;
10      else if (madd==2'b01)
11          y=b;
12      else if (madd==2'b10)
13          y=c;
14      else
15          y=0;
16  end
17 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: 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'.

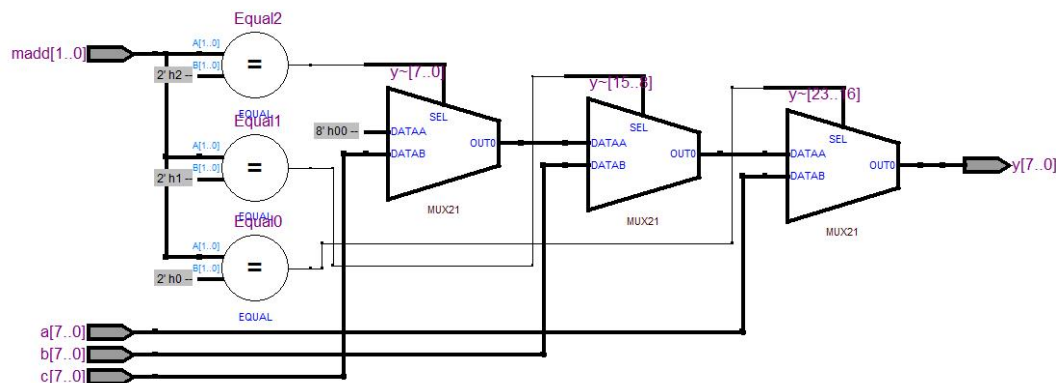
资源消耗：

```

Flow Status                Successful - Sun Nov 20 13:48:48 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        16 / 4,608 ( < 1 % )
    Total combinational functions  16 / 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



视图分析及结论：

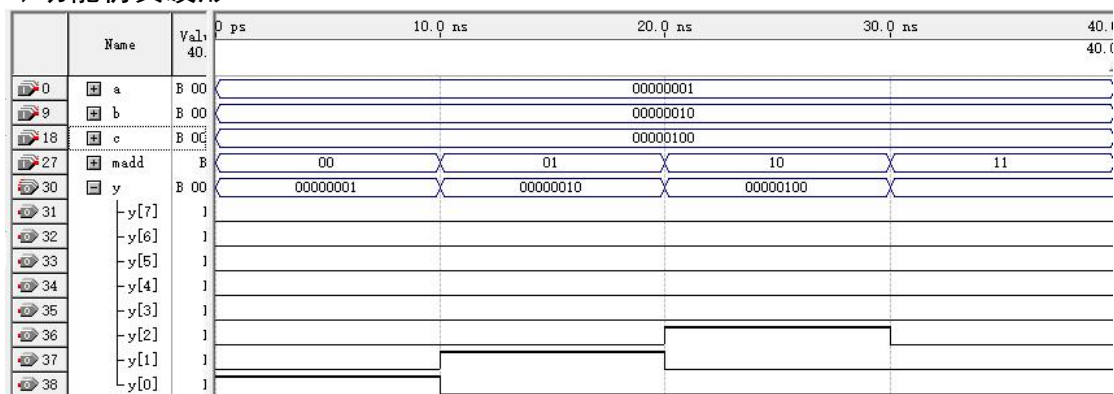
当输入为 00 时，第三个复用器 DATAB 打开，允许 a 信号输入，最终输出 a 数据。

当输入为 01 时，第二个复用器 DATAB 打开，允许 b 信号输入；第三个复用器 DATAA 打开，允许第二个复用器传递的 b 通过,最终输出 b 数据。

当输入为 10 时，第一个复用器 DATAB 打开，允许 c 信号输入；第二个复用器 DATAA 打开，允许第一个复用器传递的 c 通过,第三个复用器 DATAA 打开，允许第二个复用器传递的 c 通过，最终输出 c 数据。

图中出现选择器，因为代码中存在条件判断，对信号进行分类判断决定输出。

E) 功能仿真波形



结果分析及结论：

madd=00 时，控制输出 y 等于 a。

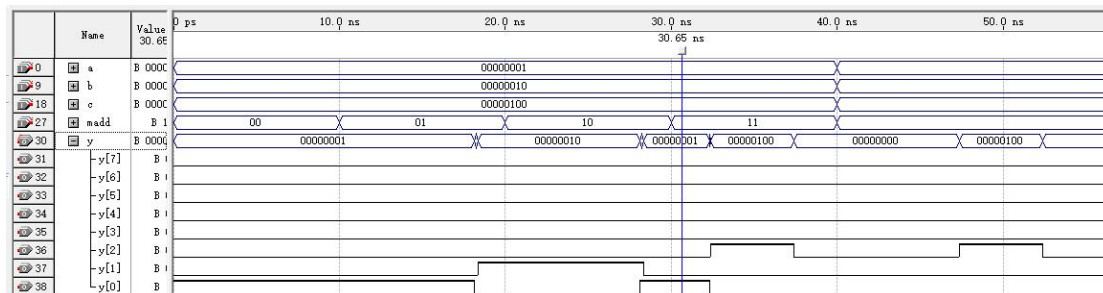
madd=01 时，控制输出 y 等于 b。

madd=10 时，控制输出 y 等于 c。

正常情况下是不会出现输入为 11 的情况的，为了避免出现锁存器，因此给输入 11 设置输出为 0。

功能仿真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



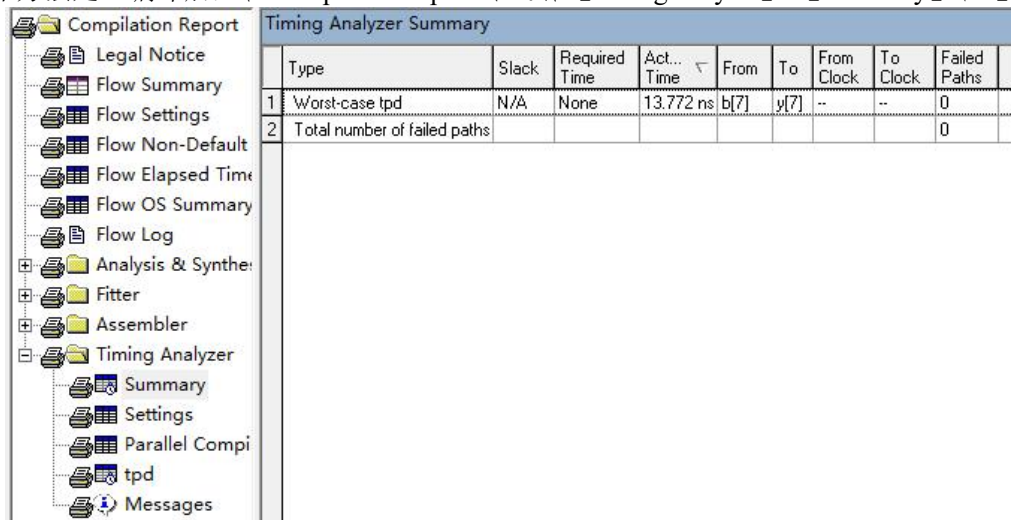
结果分析及结论：

时序仿真的输出信号有约 12ns 的延迟。

由于输入的信号的瞬间变化和路径不等长以及延时导致产生冒险，会出现一些不正确的尖峰信号，也就是毛刺现象，导致输出结果并未与预期结果相同。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。

G) 时序分析

操作方法是：编译后，在 compilationreport 中选择【timinganalysis】-【summary】和【tpd】



<ul style="list-style-type: none"> Compilation Report Legal Notice Flow Summary Flow Settings Flow Non-Default Flow Elapsed Time Flow OS Summary Flow Log Analysis & Synthesis Fitter Assembler Timing Analyzer <ul style="list-style-type: none"> Summary Settings Parallel Compil tpd Messages 	tpd					
		Slack	Required P2P Time	Actual P2P Time	From	To
	1	N/A	None	13.772 ns	b[7]	y[7]
	2	N/A	None	13.682 ns	a[7]	y[7]
	3	N/A	None	13.531 ns	a[2]	y[2]
	4	N/A	None	13.354 ns	b[1]	y[1]
	5	N/A	None	13.152 ns	c[2]	y[2]
	6	N/A	None	12.997 ns	b[2]	y[2]
	7	N/A	None	12.808 ns	madd[1]	y[1]
	8	N/A	None	12.686 ns	madd[1]	y[7]
	9	N/A	None	12.679 ns	b[3]	y[3]
	10	N/A	None	12.675 ns	a[1]	y[1]
	11	N/A	None	12.592 ns	a[4]	y[4]
	12	N/A	None	12.591 ns	a[3]	y[3]
	13	N/A	None	12.518 ns	b[5]	y[5]
	14	N/A	None	12.440 ns	b[4]	y[4]
	15	N/A	None	12.387 ns	madd[1]	y[6]
	16	N/A	None	12.382 ns	a[5]	y[5]
	17	N/A	None	12.365 ns	madd[1]	y[5]
	18	N/A	None	12.347 ns	madd[1]	y[2]
	19	N/A	None	12.328 ns	madd[1]	y[0]
	20	N/A	None	12.229 ns	a[6]	y[6]
	21	N/A	None	12.085 ns	madd[1]	y[3]
	22	N/A	None	12.059 ns	c[3]	y[3]
	23	N/A	None	11.911 ns	c[5]	y[5]
	24	N/A	None	11.863 ns	b[6]	y[6]
	25	N/A	None	11.784 ns	c[6]	y[6]
	26	N/A	None	11.752 ns	c[1]	y[1]
	27	N/A	None	11.672 ns	c[7]	y[7]
	28	N/A	None	11.625 ns	madd[1]	y[4]
	29	N/A	None	10.906 ns	c[4]	y[4]
	30	N/A	None	8.750 ns	madd[0]	y[7]
	31	N/A	None	8.679 ns	madd[0]	y[2]
	32	N/A	None	8.352 ns	madd[0]	y[1]
	33	N/A	None	8.262 ns	b[0]	y[0]
	34	N/A	None	8.197 ns	a[0]	y[0]
	35	N/A	None	8.129 ns	madd[0]	y[5]
	36	N/A	None	8.103 ns	madd[0]	y[0]
	37	N/A	None	7.933 ns	madd[0]	y[6]
	38	N/A	None	7.692 ns	madd[0]	y[4]
	39	N/A	None	7.613 ns	madd[0]	y[3]
	40	N/A	None	7.051 ns	c[0]	y[0]

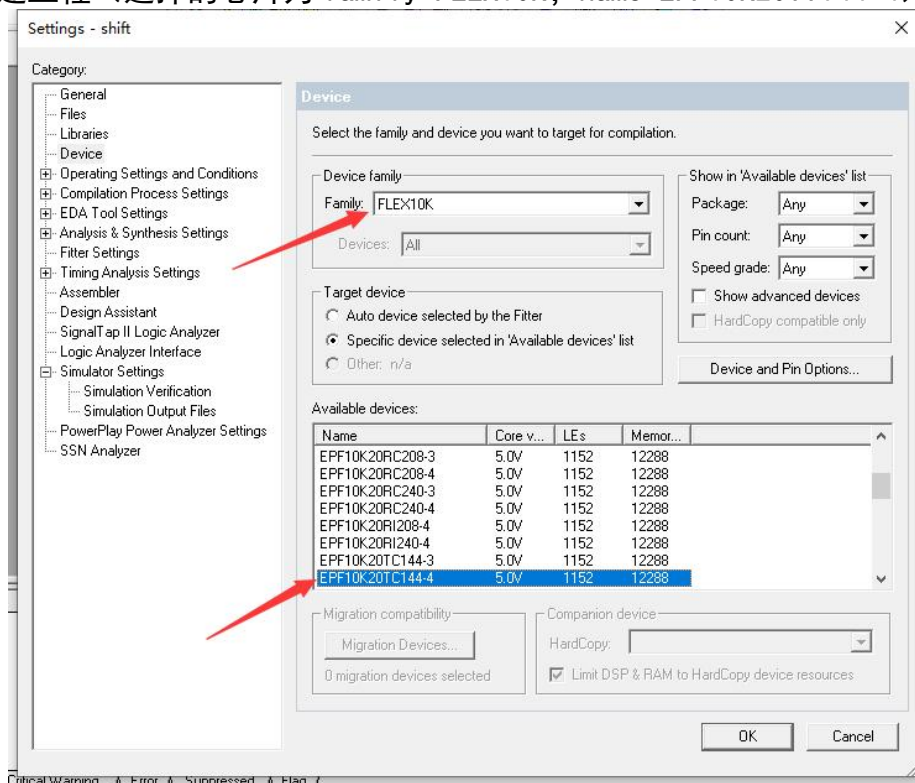
结果分析及结论：

延时范围为 7.051~13.772ns,且集中在 11-12ns 之间，与时序仿真波形图观察到的结果基本符合，结果由耗时最长的那个决定，故整体耗时为 13.772ns。

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）



B) 编写源代码

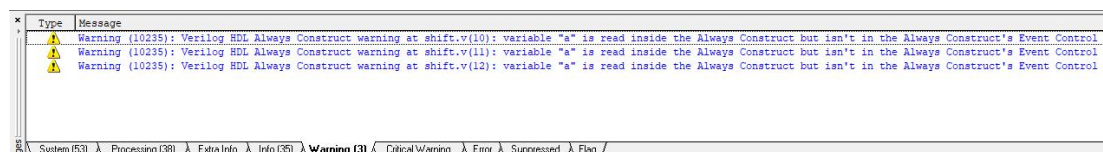
```

1 module shift(fbush, flbus, frbus, a, w, cf);
2   input fbush, flbus, frbus;
3   input [7:0] a;
4   output reg [7:0] w;
5   output reg cf;
6
7   always @(fbush, flbus, frbus, w, cf)
8   begin
9     cf=0;
10    if(fbush==1) w=a;
11    else if(flbus==1) begin w[7:1]=a[6:0]; w[0]=a[7]; cf=a[7]; end
12    else if(frbus==1) begin w[7]=a[0]; w[6:0]=a[7:1]; cf=a[0]; end
13    else w=8'hZZ;
14  end
15 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）

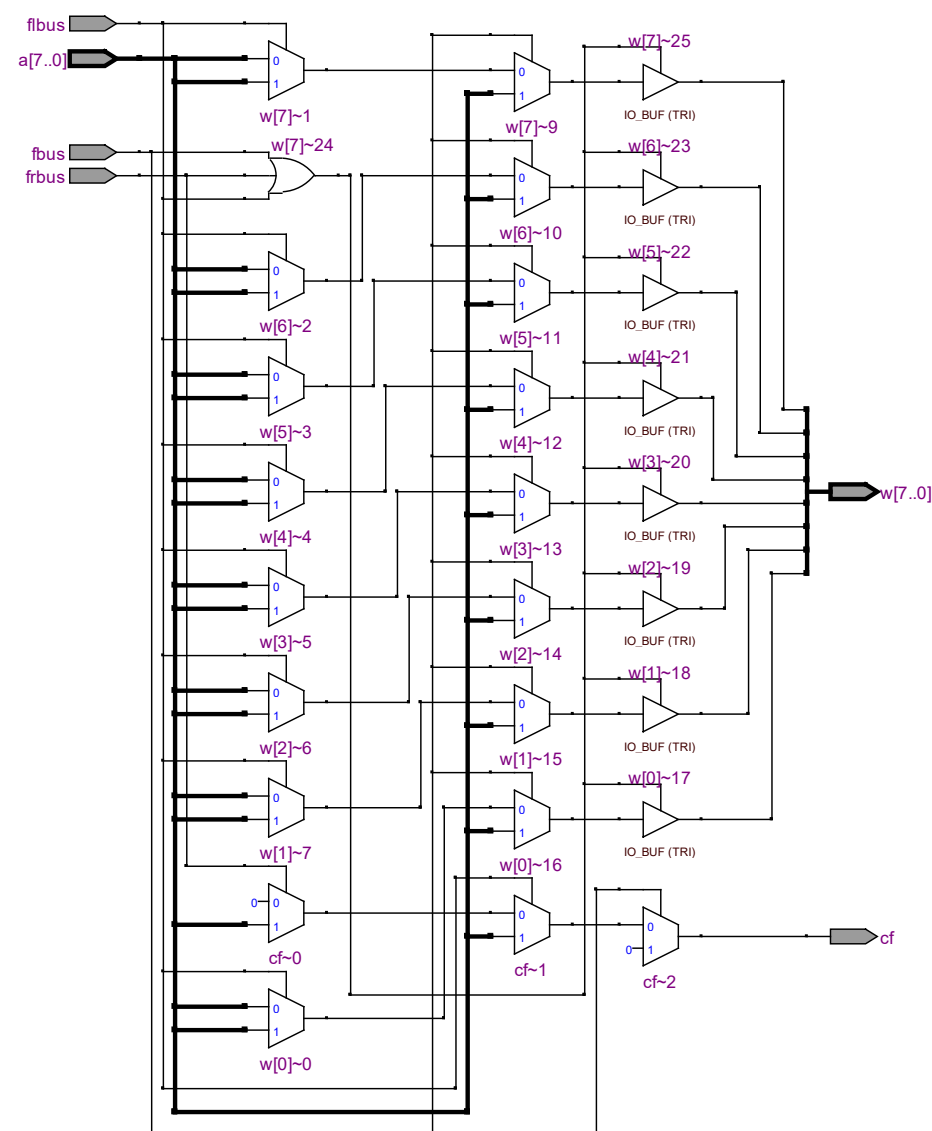
警告信息：



资源消耗:

Flow Status	Successful - Sun Nov 20 13:45:57 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	EPF10K20TC144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	19 / 1,152 (2 %)
Total pins	20 / 102 (20 %)
Total memory bits	0 / 12,288 (0 %)

D) RTL 视图

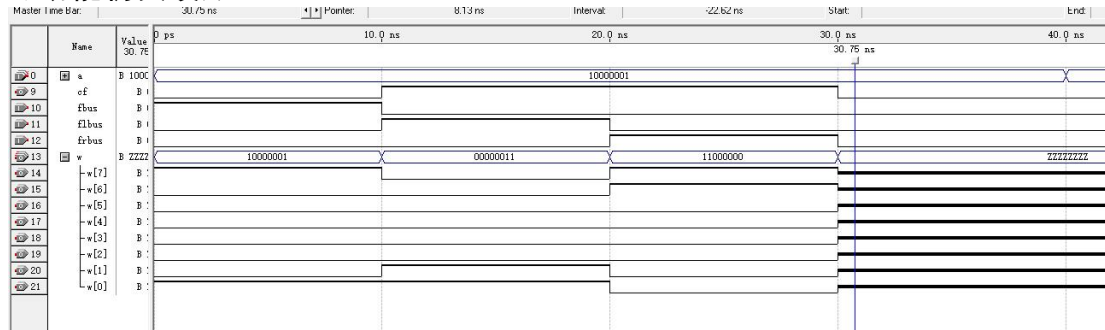


结果分析及结论：

输入信号为 fbus, frbus, flbus 和 a, 输出信号为 w。

存在多个选择器，因为是多位赋值。

E) 功能仿真波形



结果分析及结论：

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

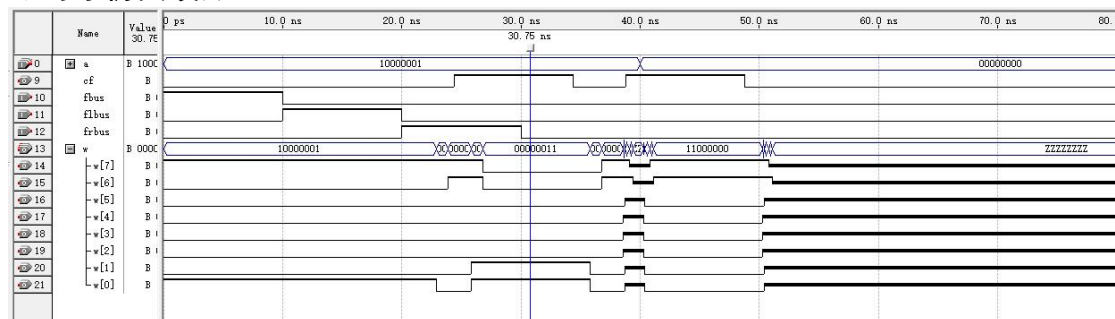
当 fbus=0, frbus=1, flbus=0, 执行右移，输出等于输入右移移位，有进位 cf 为 1

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

当控制信号对应有有效时，对应的赋值操作完成；当控制信号均低电平时，w=8'hzz，符合设计。

F) 时序仿真波形



结果分析及结论：

时序仿真存在约 18ns 的延迟。

由于输入的信号的瞬间变化，由于路径不等长以及延时导致产生冒险，会出现许多不正确的尖峰信号，也就是毛刺现象，导致输出结果并未与预期结果相同。

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	21.100 ns	fibus	w[6]	--	--	0	
2	Total number of failed paths								0	

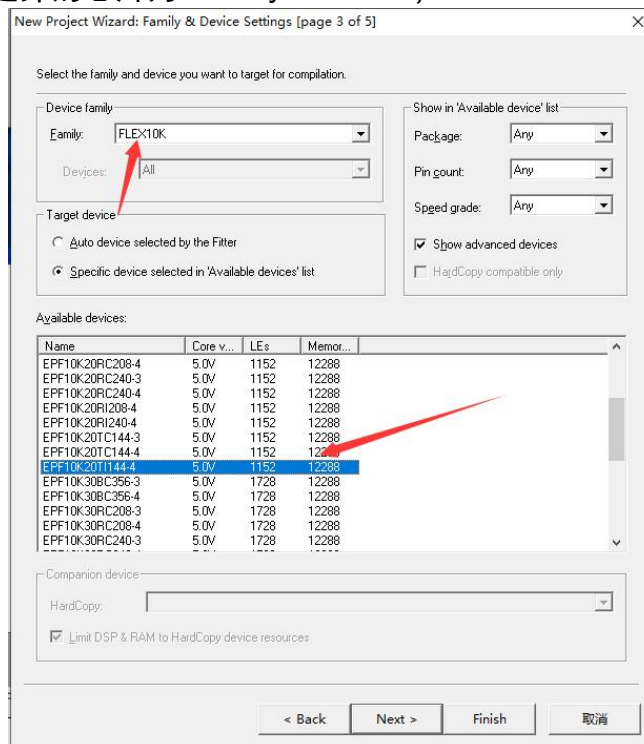
tpd					
	Slack	Required P2P Time	Actual P2P Time	From	To
4	N/A	None	20.400 ns	fibus	w[1]
5	N/A	None	20.400 ns	fibus	w[0]
6	N/A	None	20.300 ns	fibus	w[4]
7	N/A	None	20.300 ns	fibus	w[3]
8	N/A	None	20.300 ns	fibus	w[2]
9	N/A	None	20.200 ns	a[6]	w[7]
10	N/A	None	20.100 ns	a[5]	w[4]
11	N/A	None	19.800 ns	a[5]	w[6]
12	N/A	None	19.600 ns	a[6]	w[5]
13	N/A	None	19.400 ns	fibus	w[6]
14	N/A	None	19.400 ns	fibus	w[6]
15	N/A	None	19.100 ns	fibus	w[7]
16	N/A	None	19.100 ns	fibus	w[7]
17	N/A	None	18.800 ns	fibus	cf
18	N/A	None	18.700 ns	fibus	w[5]
19	N/A	None	18.700 ns	fibus	w[5]
20	N/A	None	18.700 ns	fibus	w[1]
21	N/A	None	18.700 ns	fibus	w[1]
22	N/A	None	18.700 ns	fibus	w[0]
23	N/A	None	18.700 ns	fibus	w[0]
24	N/A	None	18.600 ns	fibus	w[4]
25	N/A	None	18.600 ns	fibus	w[4]
26	N/A	None	18.600 ns	fibus	w[3]
27	N/A	None	18.600 ns	fibus	w[3]
28	N/A	None	18.600 ns	fibus	w[2]
29	N/A	None	18.600 ns	fibus	w[2]
30	N/A	None	18.200 ns	a[4]	w[5]
31	N/A	None	18.200 ns	a[3]	w[4]
32	N/A	None	18.200 ns	a[4]	w[3]
33	N/A	None	18.200 ns	a[3]	w[2]
34	N/A	None	17.300 ns	a[6]	w[6]
35	N/A	None	17.200 ns	a[5]	w[5]
36	N/A	None	17.100 ns	a[0]	cf
37	N/A	None	17.100 ns	a[0]	w[7]
38	N/A	None	17.100 ns	a[7]	w[6]
39	N/A	None	16.300 ns	a[2]	w[3]
40	N/A	None	16.300 ns	a[1]	w[2]
41	N/A	None	16.300 ns	a[0]	w[1]
42	N/A	None	16.300 ns	a[2]	w[1]
43	N/A	None	16.300 ns	a[7]	w[0]
44	N/A	None	16.300 ns	a[1]	w[0]
45	N/A	None	15.300 ns	a[4]	w[4]
46	N/A	None	15.300 ns	a[3]	w[3]
47	N/A	None	14.400 ns	fibus	cf
48	N/A	None	14.200 ns	a[7]	cf
49	N/A	None	14.200 ns	a[7]	w[7]
50	N/A	None	13.900 ns	fibus	cf
51	N/A	None	13.400 ns	a[2]	w[2]
52	N/A	None	13.400 ns	a[1]	w[1]
53	N/A	None	13.400 ns	a[0]	w[0]

结果分析及结论：

延时范围为 13.400~21.100ns,且集中在 17-18ns 之间，与时序仿真波形图观察到的结果基本符合。结果由耗时最长的那个决定，故整体耗时为 21.100ns。

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 (movb, movc, add, sub, andl, notl, rsl, jmp, jz, z, jc, c, inl, outl, nop, halt, ir, sm,
2 pc_ld, pc_inc,
3 madd,
4 ram_xl, ram_dl,
5 ir_ld,
6 reg_ra, reg_wa, reg_we,
7 alu_m, alu_s,
8 shi_fbus, shi_flbus, shi_frbus,
9 cf_en, zf_en, sm_en, in_en, out_en);
10
11 input mova, movb, movc, add, sub, andl, notl, rsl, jmp, jz, z, jc, c, inl, outl, nop, halt, ir, sm;
12 input [7:0] ir;
13 output reg [1:0] reg_ra, reg_wa, madd;
14 output reg [3:0] alu_s;
15 output reg pc_ld, pc_inc,
16 reg_we, ram_xl, ram_dl, alu_m,
17 shi_fbus, shi_flbus, shi_frbus,
18 ir_ld, cf_en, zf_en, sm_en, in_en, out_en;
19 always@ (mova, movb, movc, add, sub, andl, notl, rsl, jmp, jz, jc, inl, outl, nop, halt, ir, sm)
20 =
21   begin
22     reg_ra=ir[1:0];
23     reg_wa=ir[3:2];
24     alu_s=ir[7:4];
25     pc_ld=jmp | (jc&c) | (jz&z);
26     pc_inc=(~sm) | (jc&(~c)) | (jz&(~z));
27     reg_we=~(sm&(mova|movc|add|sub|andl|notl|l|rsr|l|inl));
28     ram_xl=movb;
29     ram_dl=(~sm) | movc | jmp | (jz&z) | (jc&c);
30     alu_m=andl | notl | add | sub | rsl | rsl | outl;
31     shi_fbus=mova | movb | add | sub | andl | notl | outl;
32     shi_flbus=rsl;
33     shi_frbus=rsl;
34     ir_ld=~sm;
35     cf_en=add | sub | rsl | rsl;
36     zf_en=add | sub;
37     sm_en=~halt;
38     in_en=inl;
39     out_en=outl;
40     if (sm==1&&movc==1) madd=2'b01;
41     else if (sm==1&&movb==1) madd=2'b10;
42     else madd=2'b00;
43   end
44 endmodule
45

```

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）

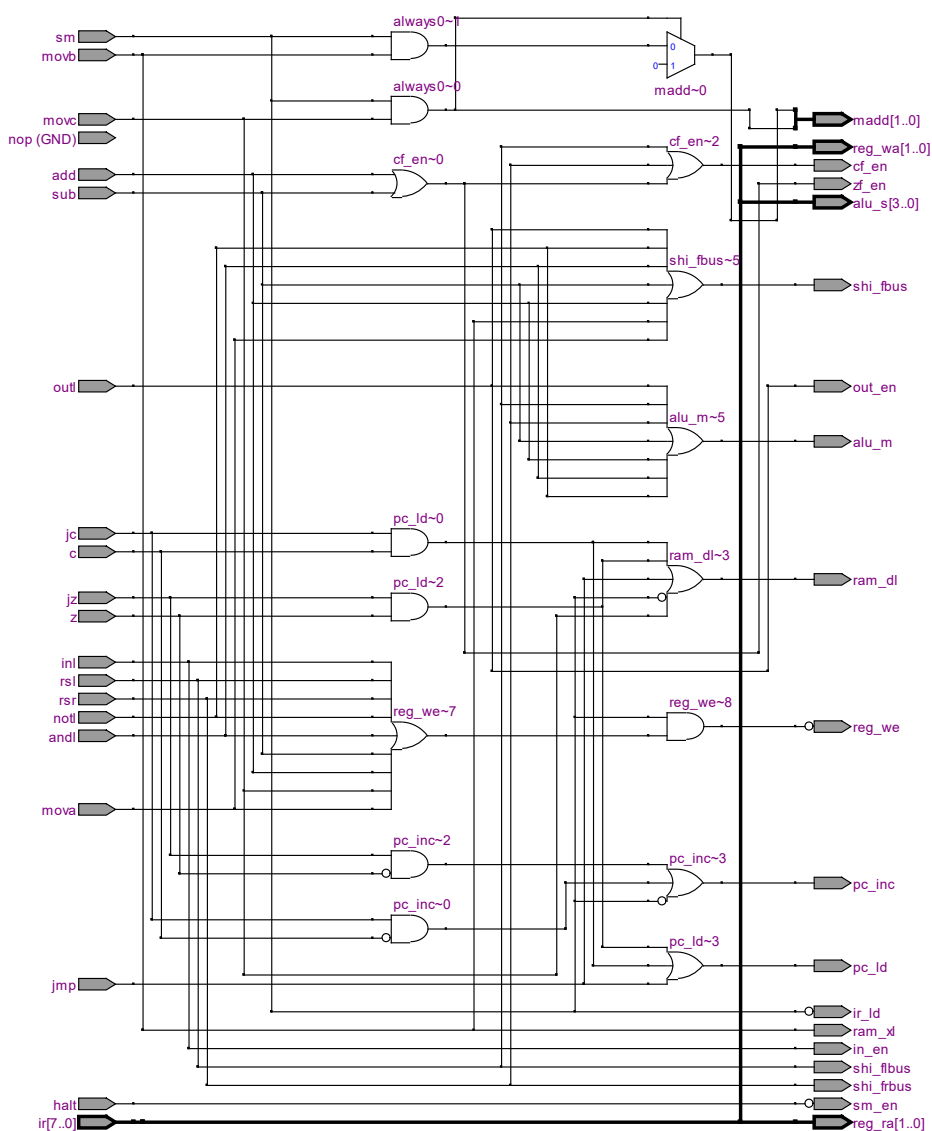
警告信息：

Type	Message
	Warning (10235): Verilog HDL Always Construct warning at con_signal.v(24): variable "c" is read inside the Always Construct but isn't in the Always Construct's Event Control
	Warning (10235): Verilog HDL Always Construct warning at con_signal.v(24): variable "z" is read inside the Always Construct but isn't in the Always Construct's Event Control
	Warning (10235): Verilog HDL Always Construct warning at con_signal.v(25): variable "c" is read inside the Always Construct but isn't in the Always Construct's Event Control
	Warning (10235): Verilog HDL Always Construct warning at con_signal.v(25): variable "z" is read inside the Always Construct but isn't in the Always Construct's Event Control
	Warning (10235): Verilog HDL Always Construct warning at con_signal.v(28): variable "z" is read inside the Always Construct but isn't in the Always Construct's Event Control
	Warning (10235): Verilog HDL Always Construct warning at con_signal.v(28): variable "c" is read inside the Always Construct but isn't in the Always Construct's Event Control
	Warning: Design contains 1 input pin(s) that do not drive logic

资源消耗：

Flow Status	Successful - Fri Nov 25 00:56:31 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	EPF10K20TII144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	30 / 1,152 (3 %)
Total pins	52 / 102 (51 %)
Total memory bits	0 / 12,288 (0 %)

D) RTL 视图



结果分析及结论：

由图验证，不存在锁存器，逻辑电路设计成功。一个元件的内部原理结构图十分复杂。

➤ halt 指令执行时, 输出全为 0

Ir 时对应得信号指令，对应着 ir 给指令 mova...赋值，完成输入

根据执行指令时各控制信号的关系:

$$\text{LD PC}=\text{JMP}+\text{JZ}\cdot\text{ZF}+\text{JC}\cdot\text{CF}$$

IN PC=/SM +JC·/CF+JZ·/ZF

MADD0=MOVC

MADD1=MOVB

$$DL = \text{SM} + \text{MOV} + \text{JMP} + \text{JZ} \cdot \text{ZF} + \text{JC} \cdot \text{CF}$$

XL=MOVB

LD IR=/SM

$$/WE=(MOVA+MOVC+ADD+SUB+AND+NOT+IN+RSR+RSL)'$$
$$RAA=MOVA+MOVB+MOVC+ADD+SUB+AND+NOT+IN+OUT+RSR+RSL$$

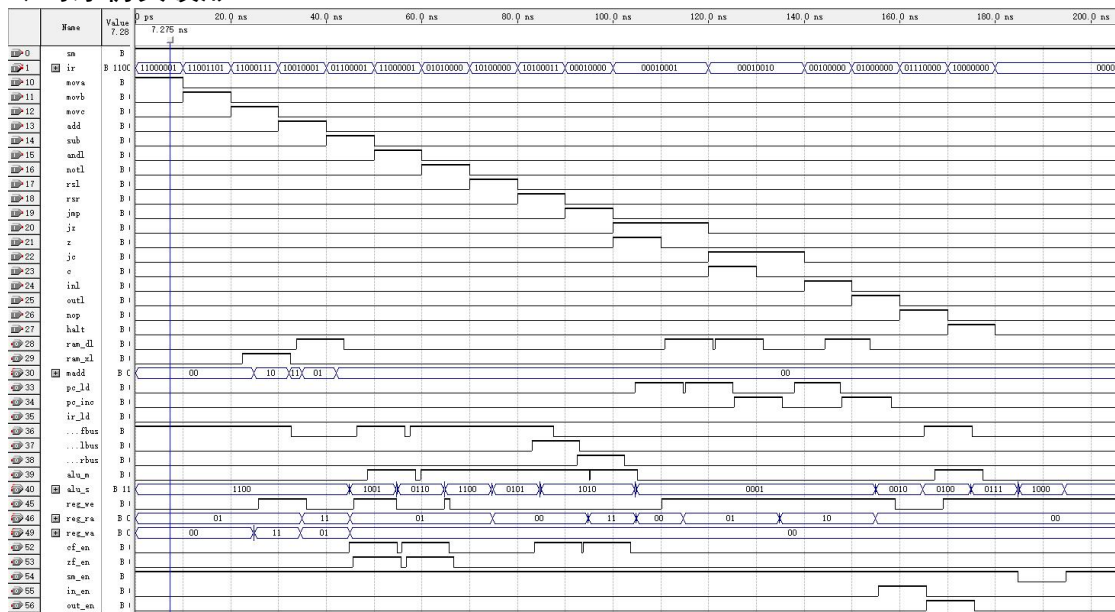
RWBA=MOVA+MOVB+MOVC+ADD+SUB+AND+NOT+IN+OUT+RSR+RSL

$$M = \text{ADD} + \text{SUB} + \text{AND} + \text{NOT} + \text{RSR} + \text{RSL}$$
$$F \rightarrow BUS = MOVA + MOVB + ADD + SUB + AND + NOT + OUT$$

FL->BUS=RSL

FR->BUS=RSR

F) 时序仿真波形



结果分析及结论:

改变输入时，先会输出会存在一段噪声，接着才输出正确的值。不同的操作对应的噪声时长也是不同的。但是总体来看，输出的值只有一小部分的时长是被噪声所掩盖的，而绝大部分时间仍然是正确输出的。

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。

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	25.800 ns	sub	reg_we	--	--	0	
2	Total number of failed paths								0	
tpd										
	Slack	Required P2P Time	Actual P2P Time	From	To					
1	N/A	None	25.800 ns	andl	reg_we					
2	N/A	None	25.800 ns	notl	reg_we					
3	N/A	None	25.800 ns	sub	reg_we					
4	N/A	None	24.700 ns	add	reg_we					
5	N/A	None	24.300 ns	jc	ram_dl					
6	N/A	None	23.800 ns	c	ram_dl					
7	N/A	None	21.400 ns	jz	ram_dl					
8	N/A	None	21.400 ns	z	ram_dl					
9	N/A	None	20.900 ns	jmp	ram_dl					
10	N/A	None	20.600 ns	rsi	reg_we					
11	N/A	None	20.200 ns	rsr	reg_we					
12	N/A	None	19.700 ns	andl	alu_m					
13	N/A	None	19.700 ns	notl	alu_m					
14	N/A	None	19.700 ns	sub	alu_m					
15	N/A	None	19.100 ns	inl	reg_we					
16	N/A	None	18.600 ns	add	alu_m					
17	N/A	None	18.300 ns	jc	pc_inc					
18	N/A	None	18.000 ns	jc	pc_ld					
19	N/A	None	17.800 ns	c	pc_inc					
20	N/A	None	17.500 ns	andl	shi_fbus					
21	N/A	None	17.500 ns	notl	shi_fbus					
22	N/A	None	17.500 ns	sub	shi_fbus					
23	N/A	None	17.500 ns	c	pc_ld					
24	N/A	None	17.300 ns	outl	alu_m					
25	N/A	None	16.700 ns	sub	zf_en					
26	N/A	None	16.400 ns	add	shi_fbus					
27	N/A	None	15.800 ns	movc	reg_we					
28	N/A	None	15.700 ns	sub	cf_en					
29	N/A	None	15.700 ns	movc	reg_we					
30	N/A	None	15.600 ns	outl	out_en					
31	N/A	None	15.600 ns	add	zf_en					
32	N/A	None	15.500 ns	inl	in_en					
33	N/A	None	15.400 ns	jz	pc_inc					
34	N/A	None	15.400 ns	z	pc_inc					
35	N/A	None	15.100 ns	outl	shi_fbus					
36	N/A	None	15.100 ns	rsr	alu_m					
37	N/A	None	15.100 ns	jz	pc_ld					
38	N/A	None	15.100 ns	z	pc_ld					
39	N/A	None	15.000 ns	rsi	alu_m					
40	N/A	None	14.900 ns	ir[4]	alu_s[0]					
41	N/A	None	14.900 ns	ir[2]	reg_wa[0]					
42	N/A	None	14.900 ns	ir[1]	reg_ra[1]					
43	N/A	None	14.900 ns	movc	madd[1]					
44	N/A	None	14.800 ns	add	cf_en					
45	N/A	None	14.800 ns	ir[6]	alu_s[2]					

46	N/A	None	14.800 ns	ir[5]	alu_s[1]
47	N/A	None	14.800 ns	movb	madd[1]
48	N/A	None	14.700 ns	halt	sm_en
49	N/A	None	14.700 ns	ir[0]	reg_ra[0]
50	N/A	None	14.600 ns	ir[3]	reg_wa[1]
51	N/A	None	14.600 ns	jmp	pc_ld
52	N/A	None	14.500 ns	ir[7]	alu_s[3]
53	N/A	None	14.400 ns	sm	madd[1]
54	N/A	None	13.700 ns	rsr	cf_en
55	N/A	None	13.700 ns	movc	ram_dl
56	N/A	None	13.500 ns	rsl	cf_en
57	N/A	None	13.400 ns	sm	reg_we
58	N/A	None	13.200 ns	sm	ram_dl
59	N/A	None	13.000 ns	rsl	shi_fibus
60	N/A	None	12.900 ns	sm	pc_inc
61	N/A	None	12.700 ns	movb	shi_fibus
62	N/A	None	12.700 ns	movb	shi_fibus
63	N/A	None	12.700 ns	sm	ir_ld
64	N/A	None	12.600 ns	sm	madd[0]
65	N/A	None	12.500 ns	movb	ram_xl
66	N/A	None	12.400 ns	rsr	shi_fibus
67	N/A	None	12.100 ns	movc	madd[0]

结果分析及结论：

延时范围为 12.100~25.800ns,且集中在 14-16ns 之间，与时序仿真波形图观察到的结果基本符合。

从 sub 到 reg_we 的最坏定时情况的 tpd 为 25.800ns,并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？

应该输出高阻。

移位逻辑不工作时，为了预防移位逻辑输出数据到总线中和其他位置输出到总线中的数据起冲突，应该阻止移位逻辑向总线输出数据，故此时移位逻辑输出应该为高阻。否则移位逻辑会将数据输送到总线，使得总线中有多组数据，这样会干扰其他的模块的正常运转。

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。

每条指令都需要两个周期完成，在控制信号的作用下，完成取指和执行操作。ADDA,B 取指令时的控制信号为：SM 为 0，INPC 为 1，LDPC 为 0,MADD 为 00,XL 为 0，DL 为 1,LDIR 为 1。执行指令时的控制信号为：SM 为 1，RAA₁RAA₀ 为 01,RWBA₁RWBA₀ 为 00,WE 为 0，M 为 0，S₃S₂S₁S₀ 为 1001,FBUS 为 1,FLBUS 和 FRBUS 都为 0。

ADD A, B的控制信号：

● 取指令（SM=0）

IN PC: 1, MADD: 00,

XL: 0, DL: 1, LD IR: 1

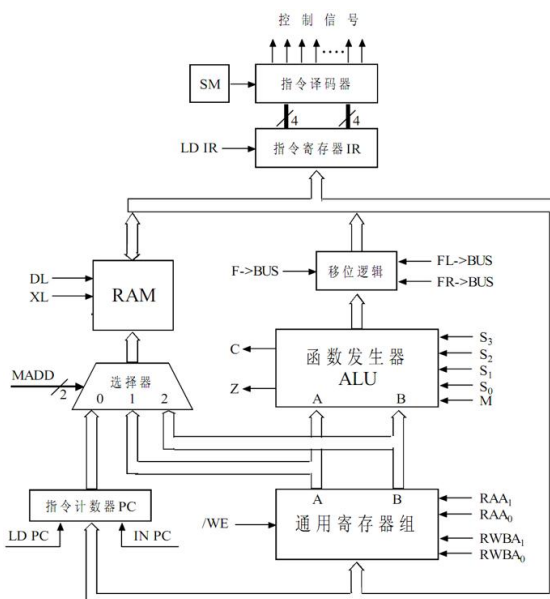
● 执行指令（SM=1）

RAA₁ RAA₀: 01RWBA₁ RWBA₀: 00

/WE: 0 M: 0,

S₃S₂S₁S₀: 1001 F→BUS:1,

FL→BUS:0, FR→BUS:0,



3. 如何产生正确的控制信号以及具体的编程实现？

首先需要对 16 种命令分别是如何控制信号的一一分析，得出不同命令下信号的输出是什么，然后根据分析的结果将控制信号的命令用或运算组合起来。其次再根据之前所列的每个控制信号的或运算式子，采用 if 语句/whenelse 语句进行编程实现。

LD PC=JMP+JZ·ZF+JC·CF

IN PC=/SM +JC·/CF+JZ·/ZF

MADD0=MOVC

MADD1=MOVB

DL=/SM+MOVC+JMP+JZ·ZF+JC·CF

XL=MOVB

LD IR=/SM

/WE=(MOVA+MOVC+ADD+SUB+AND+NOT+IN+RSR+RSL)'

RAA=MOVA+MOVB+MOVC+ADD+SUB+AND+NOT+IN+OUT+RSR+RSL

RWBA=MOVA+MOVB+MOVC+ADD+SUB+AND+NOT+IN+OUT+RSR+RSL

M= ADD+SUB+AND+NOT+RSR+RSL

F->BUS=MOVA+MOVB+ADD+SUB+AND+NOT+OUT

FL->BUS=RSL

FR->BUS=RSR

编程结果如下：

```
always@(movb, movb, movc, add, sub, andl, notl, rsl, jmp, jz, jc, inl, outl, nop, halt, ir, sm)
begin
    reg_ra=ir[1:0];
    reg_wa=ir[3:2];
    alu_s=ir[7:4];
    pc_ld=jump|(jc&c)|(jz&z);
    pc_inc=(~sm)|(jc&(~c))|(jz&(~z));
    reg_we=~(sm&(movb|movc|add|sub|andl|notl|rsl|inl));
    ram_xl=movb;
    ram_dl=(~sm)|movc|jump|(jz&z)|(jc&c);
    alu_m=andl|notl|add|sub|rsl|outl;
    shi_fbus=movb|movb|add|sub|andl|notl|outl;
    shi_fbus=rsl;
    shi_fbus=rsl;
    ir_ld=~sm;
    cf_en=add|sub|rsl|rsl;
    zf_en=add|sub;
    sm_en=~halt;
    in_en=inl;
    out_en=outl;
    if(sm==1&&movc==1) madd=2'b01;
    else if(sm==1&&movb==1) madd=2'b10;
    else madd=2'b00;
end
```

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。

- 掌握的理论：8重3-1多路复用器、移位逻辑、控制信号的逻辑结构，模型机的内部结构和工作原理。
- 遇到的困难：在运行调试过程中的锁存器问题
- 解决的办法：讨论+问老师
- 经验教训：在实现代码前，先认真看一遍实验手册，确保正确性。正确理解工程要实现的功能就可以将代码简化到最佳。

2、对本实验内容、过程和方法的改进建议（可选项）。