

模型机设计报告

班级 计科 2003 姓名 袁鹏 学号 202008010321

一、设计目的

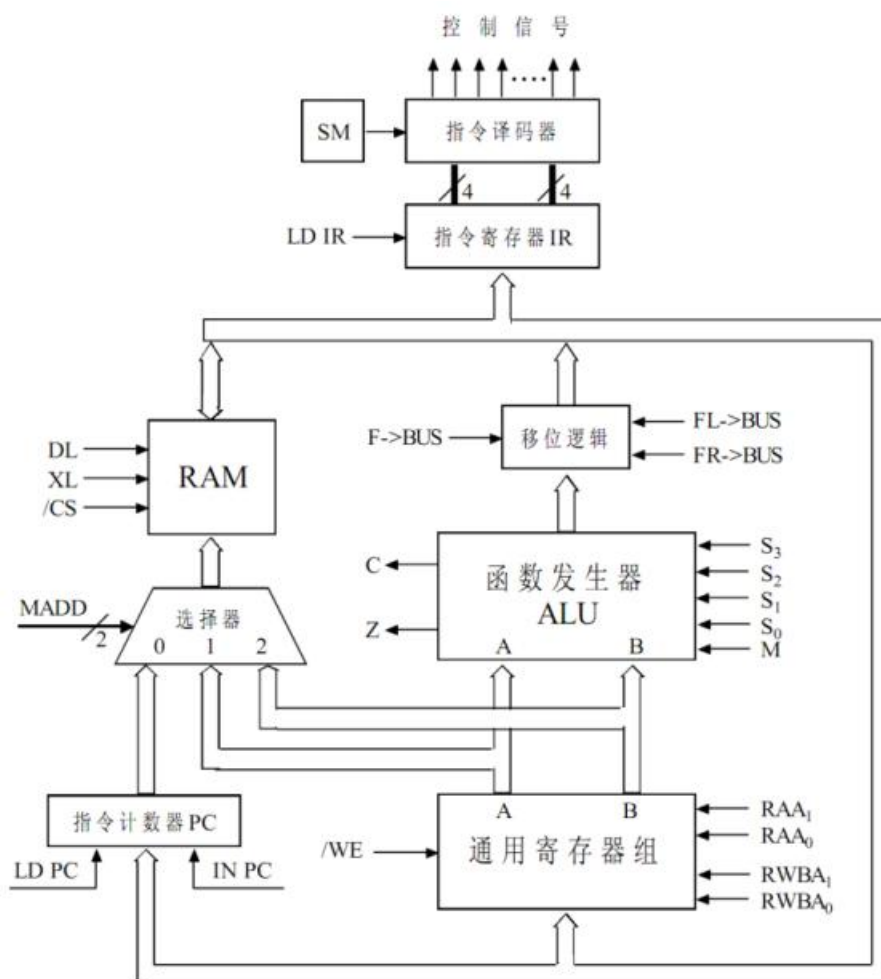
完整、连贯地运用《电路与电子学》所学到的数电知识，熟练掌握现代 EDA 工具基本使用方法，为后续课程学习和今后从事相关工作打下良好的基础或做下一些铺垫。

二、设计内容

- 1、按照给定的数据通路、数据格式和指令系统，使用 EDA 工具设计一台用硬连线逻辑控制的简易计算机；
- 2、整理出设计报告。

三、详细设计

3.1 设计的整体架构



3.2 各模块的具体实现

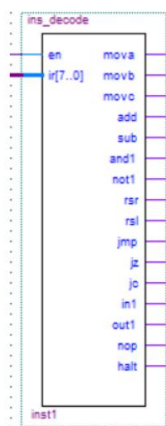
(此部分必须有模块的接口设计，功能实现，功能的仿真验证等内容。)

1. 指令译码器

指令译码器是根据指令系统表中的指令编码，对输入的 8 位指令进行解析，判定是哪条指令，则对应指令的输出为 1，否则输出为 0。

表 1 指令系统表

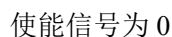
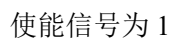
汇编符号	功能	编码
MOV R1, R2	$(R2) \rightarrow R1$	1100 R1 R2
MOV M, R2	$(R2) \rightarrow (C)$	1100 11 R2
MOV R1, M	$((C)) \rightarrow R1$	1100 R1 11
ADD R1, R2	$(R1) + (R2) \rightarrow R1$	1001 R1 R2
SUB R1, R2	$(R1) - (R2) \rightarrow R1$	0110 R1 R2
AND R1, R2	$(R1) \& (R2) \rightarrow R1$	1011 R1 R2
NOT R1	$\neg (R1) \rightarrow R1$	0101 R1 XX
RSR R1	$(R1)$ 循环右移一位 $\rightarrow R1$	1010 R1 00
RSL R1	$(R1)$ 循环左移一位 $\rightarrow R1$	1010 R1 11
JMP add	$add \rightarrow PC$	0011 00 00, address
JZ add	结果为 0 时 $add \rightarrow PC$	0011 00 01, address
JC add	结果有进位时 $add \rightarrow PC$	0011 00 10, address
IN R1	$(\text{开关 } 7-0) \rightarrow R1$	0010 R1 XX
OUT R1	$(R1) \rightarrow \text{发光二极管 } 7-0$	0100 R1 XX
NOP	$(PC) + 1 \rightarrow PC$	0111 00 00
HALT	停机	1000 00 00



指令译码器的输入输出引脚如上图所示。en 为使能信号，ir[7..0] 是 8 位指令编码，输出是对应的 16 条指令。引脚之间的相互关系如下表所示：

表 2 指令译码器引脚关系

en	ir[7..0]	16 个输出信号
1	8 位的指令编码	指令编码对应的指令输出为 1，其它输出为 0
0	8 位的指令编码	不管 ir 为何值，16 个输出全为 0



2. ALU

ADD R1, R2

SUB R1, R2

AND R1, R2

NOT R1

这类指令的执行过程为:

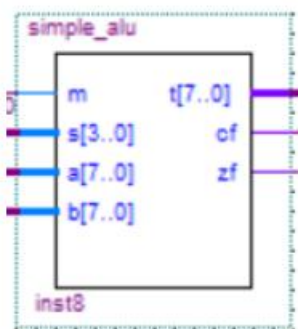
第 3 页 共 17 页

R1 的编码通过 RWBA1、RWBA0 从通用寄存器组 D 口读出 R1 的内容，在 S3~S0 和 M 的控制下，实现运算，经移位逻辑送入总线 BUS；由 /WE 控制和 R1 的编码选择 RWBA1、RWBA0，将 BUS 上的数据写入通用寄存器 R1。其中 ADD 和 SUB 指令影响状态位 Cf 和 Zf。

指令具体功能如下：

汇编符号	功能	编码
ADD R1, R2	$(R1) + (R2) \rightarrow R1$	1001 R1 R2
SUB R1, R2	$(R1) - (R2) \rightarrow R1$	0110 R1 R2
AND R1, R2	$(R1) \& (R2) \rightarrow R1$	1011 R1 R2
NOT R1	$\neg (R1) \rightarrow R1$	0101 R1 XX

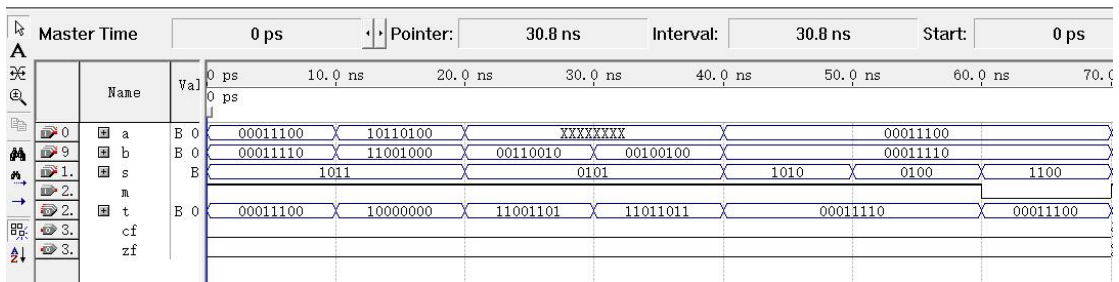
ALU 除了要完成 ADD、SUB、AND、NOT 运算外，还需在 MOVA、MOVB、RSR、RSL 和 OUT 五条指令执行时，提供将数据传送至总线的通路。ALU 模块的输入输出引脚如下图所示：



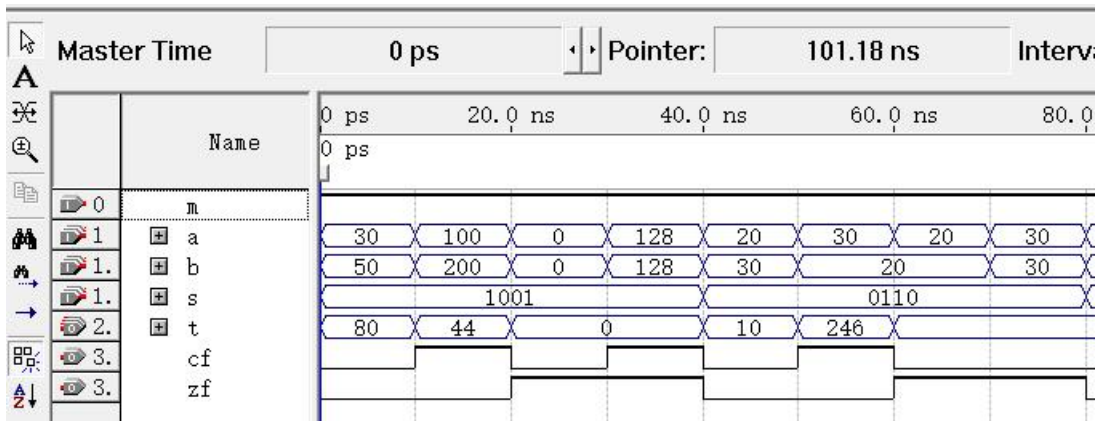
其中 m 和 s[3..0] 是控制信号，控制 a[7..0] 和 b[7..0] 输入的数据进行什么操作，并将产生的结果输出到 t[7..0]、cf 和 zf。各引脚间的相互关系如下表所示：

表 3 ALU 引脚关系

m	s[3..0]	t[7..0]	cf	zf
1	1001	$t=a+b$	有进位, cf=1 无进位, cf=0	和为零, zf=1 和不为零, zf=0
1	0110	$t=b-a$	有借位, cf=1 无借位, cf=0	差为零, zf=1 差不为零, zf=0
1	1011	$t=a\&b$	不影响	不影响
1	0101	$t=\neg b$ (注: b 相反)	不影响	不影响
1	1010 或 0100	$t=b$	不影响	不影响
0	1100	$t=a$	不影响	不影响



与运算、取反、传输通道的仿真波形



加法、减法的仿真波形

当 cf 和 zf 没有输出时，为了避免产生锁存器，默认 cf 和 zf 输出 0。由于存储 cf 和 zf 的寄存器使能信号处于非使能状态，故不用担心输出的 cf 和 zf 被写入进寄存器。输入 a 是源寄存器口，b 是目的寄存器口，执行取反操作时，目的寄存器口输入的数据有效，即对 b 取反。进行减法操作是目的寄存器的值减去源寄存器的值，即 $b-a$ 。

3. 多路复用器

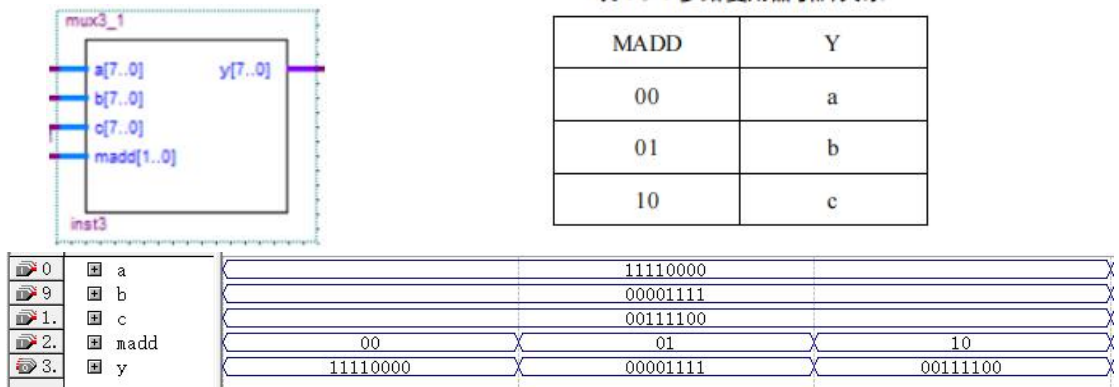
多路复用器是一个组合电路，它可以从多个输入中选择一个输入，并将信息直接传输到输出。选择哪一条输入线由一组输入变量控制，它们被称为选择输入。通常， 2^n 条输入线要 n 个选择输入，选择输入的位组合决定选择哪个输入线。

8 重 3-1 多路复用器有 3 个输入 1 个输出，每个输入和输出都是 8 位，所以称之为 8 重，MADD 选择将哪个输入传至输出，8 重 3-1 多路复用器的引脚如图所示：

引脚之间的相互关系如下表所示：

表 1 3-1 多路复用器引脚关系

MADD	Y
00	a
01	b
10	c



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。

4. 移位逻辑

移位指令：

RSR R1

RSL R1

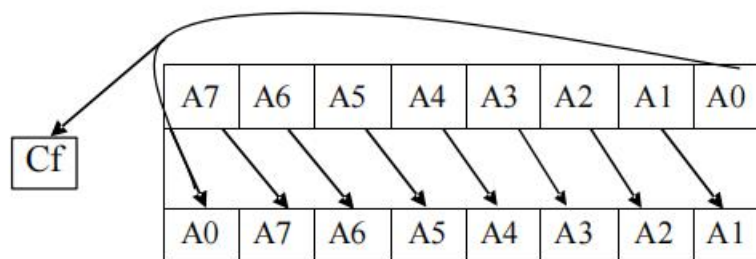
这类指令的执行过程为：

由 R1 的编码通过 RWBA1、RWBA0 从通用寄存器组 D 口读出 R1 的内容，在 S3~S0 和 M 的控制下通过 ALU，经移位逻辑循环右移或循环左移后送入总线 BUS；再由 /WE 控制和 R1 的编码选择 RWBA1、RWBA0，将 BUS 上的数据写入通用寄存器 R1。

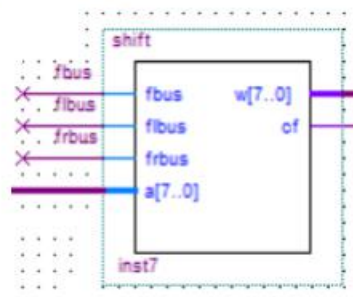
指令具体功能如下：

汇编符号	功能	编码
RSR R1	(R1) 循环右移一位 → R1	1010 R1 00
RSL R1	(R1) 循环左移一位 → R1	1010 R1 11

移位逻辑要实现 RSR、RSL 操作，还在 MOVA、MOVB、ADD、SUB、AND、NOT、OUT 指令执行时，将数据传输至 BUS 总线。RSR 循环移位操作如下：



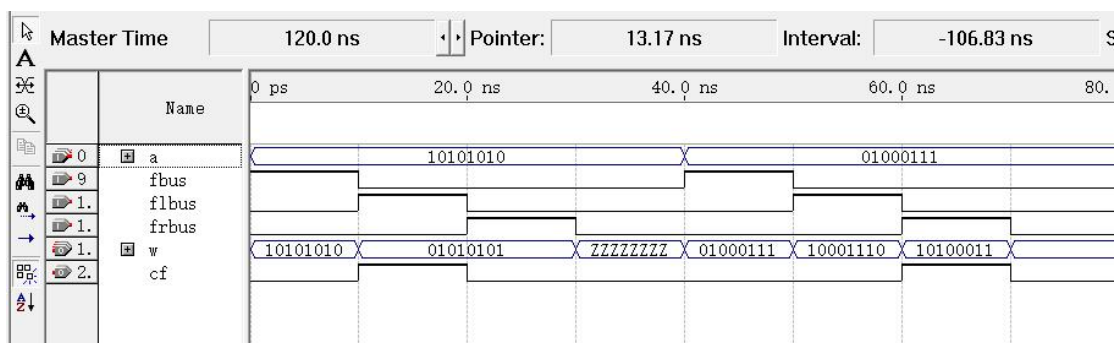
移位逻辑的输入输出引脚如下图所示：



引脚之间的相互关系如下表所示：

表 2 移位逻辑引脚关系

F_BUS	FL_BUS	FR_BUS	W	C _f
1	0	0	$w \leftarrow a$	不影响
0	1	0	$w \leftarrow a[6:0], a[7]$	$a[7]$
0	0	1	$w \leftarrow a[0], a[7:1]$	$a[0]$
0	0	0	8'hZZ	不影响

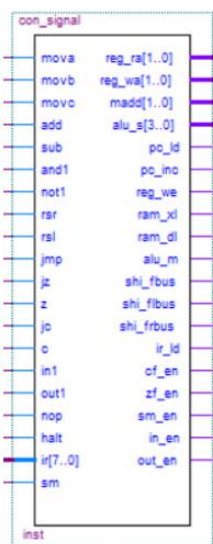


结果分析及结论：

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。

5. 控制信号产生逻辑

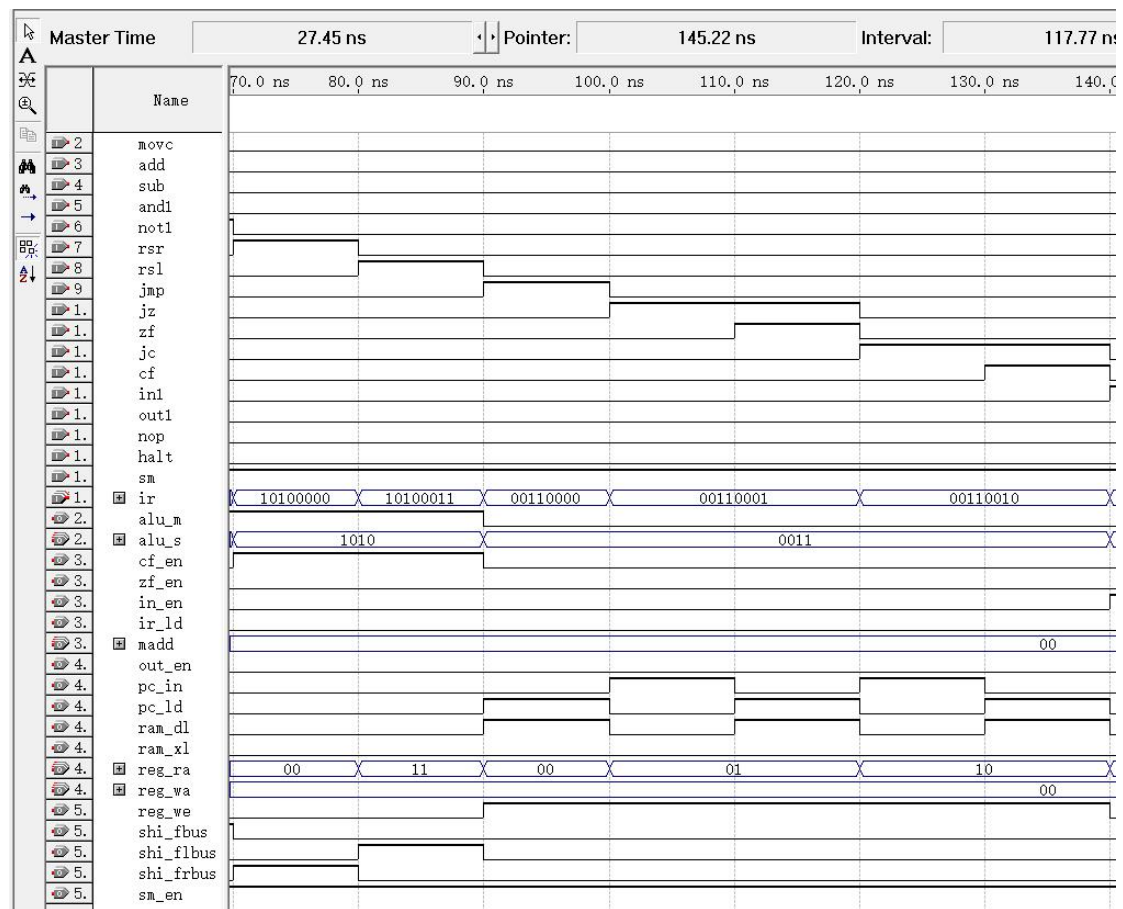
控制信号产生逻辑接收指令译码器的输出，在 SM、IR[7..0]以及状态位 Cf 和 Zf 的配合下产生每个模块所需要的控制信号。



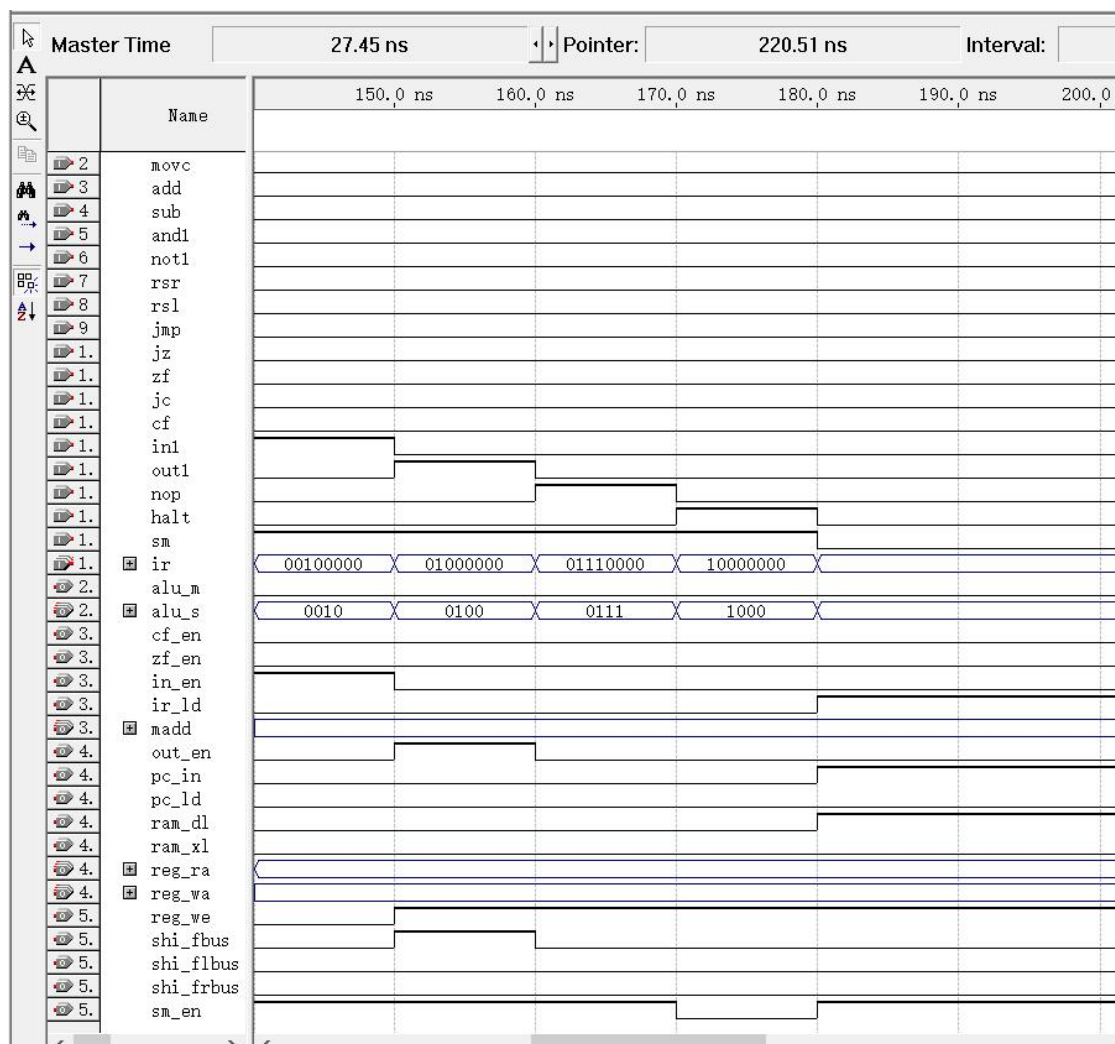
控制信号产生逻辑的输入输出引脚如下图所示：



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：

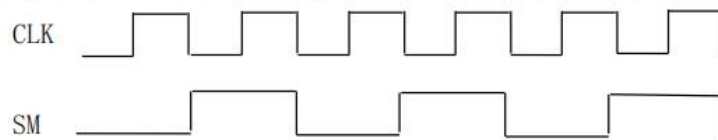
三个传送类指令涉及到数据的源端和目的端，所以需要控制 madd，ram 的读写，通用寄存器组的写，移位逻辑通道的通和不通。4 个算术逻辑运算类指令和 2 个移位逻辑指令需要从通用寄存器组读和写数据，需要通过数据通路，所以要发出数据通路相关的控制信号。三个转移指令则涉及是否到从 ram 中取指令送到指令计数器中，jmp 是一定会取出指令送到指令计数器，jz 和 jc 则实际是否成功，若不成功则 pc+1，否则与 jmp 一致。out 和 in 指令则与外界输入设备相连，可以通过输入输出设备控制 cpu 的运行，或者查看 cpu 内的数据。执行 nop 指令时不进行任何操作，而执行 halt 停机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

6. SM

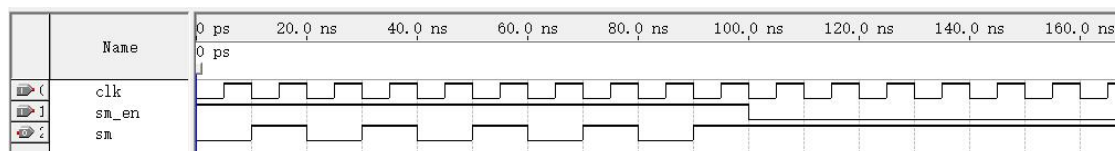
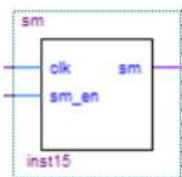
模型机中所有指令都要求两个周期完成，其中一个周期取指令，一个周期执行指令。如何区分当前周期是取指令还是执行指令呢？这就需要 SM 配合。SM 为 0

是取指令周期；SM 为 1 是执行指令周期。SM 的功能及封装如下：



SM 的功能表：

CLK	SM_EN	功能
	1	SM \leftrightarrow SM取反



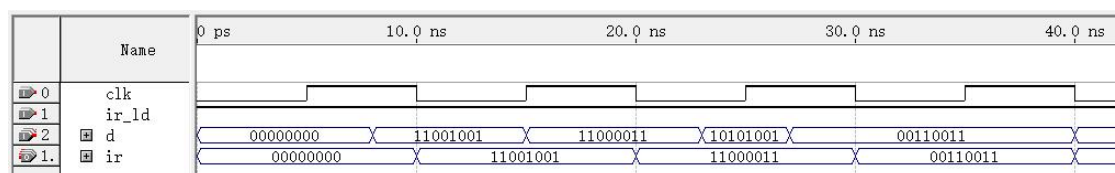
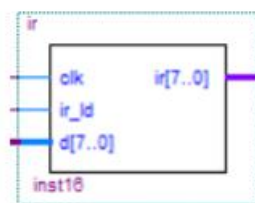
6. 指令寄存器 IR

指令寄存器（IR，Instruction Register）用于暂存当前正在执行的指令。指令寄存器将总线送来的指令存入 8 位寄存器中，但并不是每次总线上的数据都需要寄存，因为数据总线上有时传输指令，有时传输数据。当控制信号 IR_LD 为 1

时，指令寄存器在时钟信号 CLK 的下降沿将总线传输的指令写入寄存器。

指令寄存器 IR 是一个 8 位寄存器。其功能及封装如下：

CLK	Ir_ld	功能
	1	d写入ir

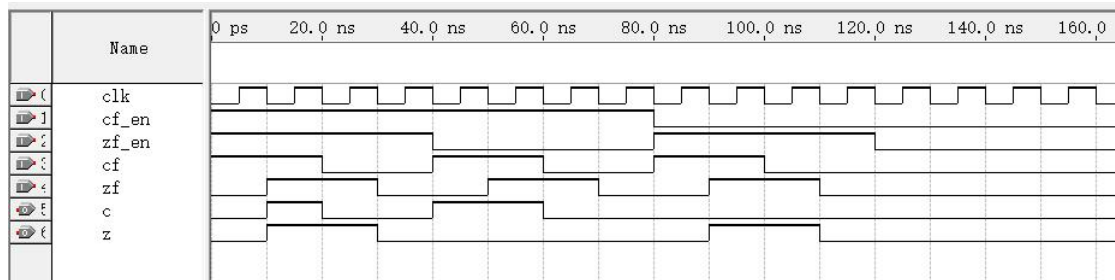
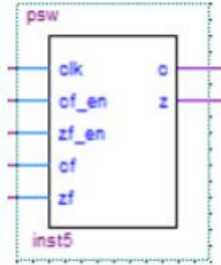


7. 状态寄存器 PSW

状态寄存器 PSW 是计算机系统的核心部件——运算器的扩展部分。本模型机

PSW 用来存放 ADD、SUB、RSR、RSL 指令执行结果的状态标志，如有无借位进位（C）、结果是否为零（Z）。有些机器也将 PSW 称为标志寄存器 FR（Flag Register）。本模型机 PSW 是一个 2 位寄存器。其功能及封装如下：

CLK	控制信号	功能
	cf_en=1	cf写入c
	zf_en=1	zf写入z



8.指令计数器 PC



指令计数器 PC 存储当前指令在 RAM 中存放的地址。

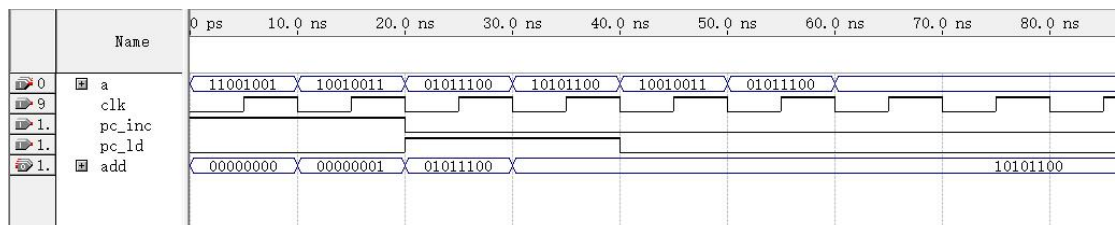
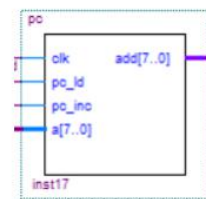
CPU 执行一条指令，根据 PC 中存放的指令地址，将指令从 RAM 读出写入指令寄存器 IR 中，此过程称为“取指令”。在每条指令读取后，指令计数器 PC 中的地址

自动加 1，指向下一条指令在 RAM 中的存放地址。跳转指令如 JMP、JZ、JC 让程序

跳转至指定地址去执行，这时 PC 需要装载跳转地址。

模型机的指令计数器 PC 是一个 8 位计数器，其的功能及封装如下：


CLK	pc_inc	pc_ld	功能
	1	0	add[7..0] 中数据自加1
	0	1	a[7..0] 写入 add[7..0]

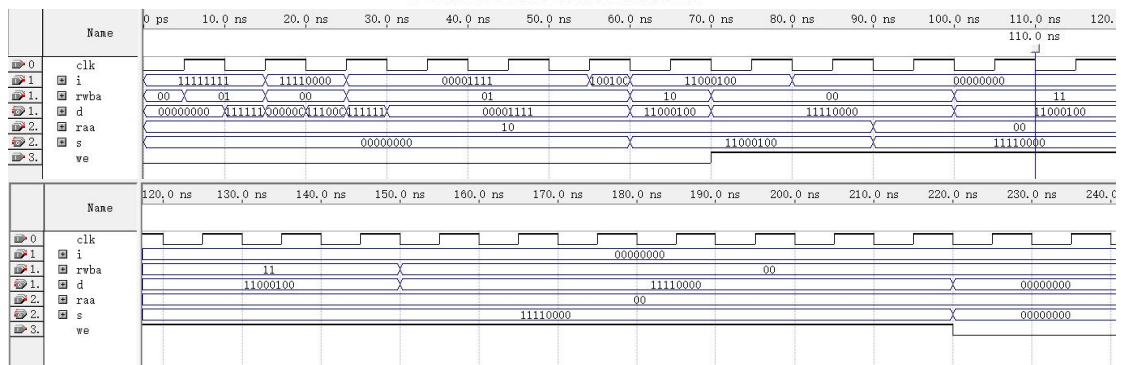
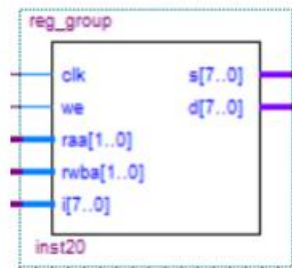


9. 通用寄存器组

寄存器主要用来保存操作数和运算结果等信息，从而节省从 RAM 中读取操作数所需占用总线 and 访问存储器的时间。

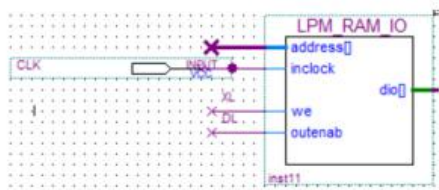
模型机的通用寄存器组包含 3 个 8 位寄存器 A、B、C，可对这 3 个寄存器进行读写操作。其功能及封装如下：

操作	CLK	WE	功能
读			根据RAA[1..0]的值从A,B,C中选择一个寄存器的值由S口输出 根据RWBA[1..0]的值从A,B,C中选择一个寄存器的值由D口输出
写		0	控制信号WE为0, 根据RWBA[1..0]的值, 在CLK下降沿将外部输入i写入A,B,C三个寄存器中的某个寄存器中。



10. RAM

半导体存储器的种类很多,从功能上可以分为只读存储器 ROM 和随机存储器 RAM 两大类。随机存储器 RAM 是与 CPU 直接交换数据的内部存储器,也叫主存(内存)。它可以随时读写,而且速度很快,通常作为操作系统或其他正在运行中的程序的临时数据存储媒介。存储元是构成存储器的存储介质,它可存储一个二进制位。由若干个存储元组成一个存储单元,然后再由许多存储单元组成一个存储器。一个存储器包含许多存储单元,每个存储单元可存放一个字节。每个存储单元的位置都有一个编号,即地址,一般用十六进制表示。一个存储器中所有存储单元可存放数据的总和称为它的存储容量。比如,一个存储器的地址码由 8 位二进制数(即 2 位十六进制数)4 组成,则可表示 2 的 8 次方,即 256 个存储单元地址,每个存储单元存放一个字节,则该存储器的存储位数为 256×8 ,即 2Kbit。



CLK	We (XL)	outenab(DL)	功能
	0	0	Dio<=高阻态Z
	1	0	Dio的数据写入address所指定的存储单元
	0	1	address所指定的存储单元数据从dio输出

Compilation Repo...

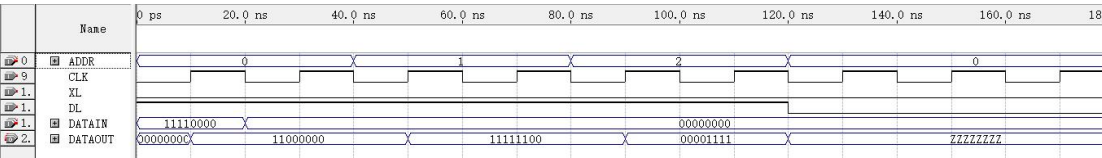
ram.bdf

RTL Viewer

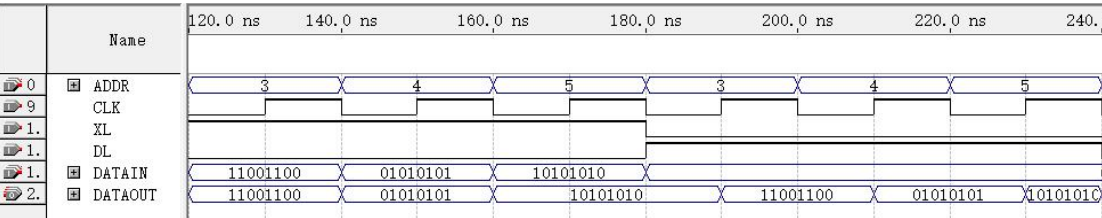
ram.mif

Addr	+0	+1	+2	+3	+4	+5	+6
00	11000000	11111100	00001111	00000000	00000000	00000000	00000000
08	00000000	00000000	00000000	00000000	00000000	00000000	00000000
10	00000000	00000000	00000000	00000000	00000000	00000000	00000000

在 mif 文件中存储 3 个数据



读取数据成功，输出高阻态成功



写入数据并读取成功

四、系统测试

4.1 测试环境

设备：荣耀 magicbook14

AMD Ryzen 7 4700U with Radeon Graphics 2.00 GHz

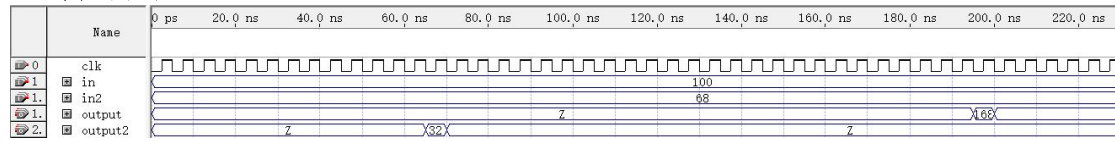
软件：Help Version: Quartus II Help Version 9.0

4.2 测试代码

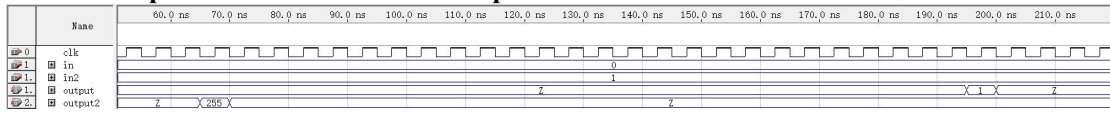
1. in 给 a:00100000
2. nop: 01110000
3. in 给 b:00100100
4. nop: 01110000
5. movb(ram \leftarrow a):11001100 (把 a 备份在 ram 中)
6. sub(a-b):01100001
7. out(a):01000000 (输出 a)
8. movc(a \leftarrow ram):11000011(把 a 还回来)
9. add(a+b):10010001 (把 a+b 给 a)
10. jc:00110010,00011000
11. jz:00110001,00011000
12. mova(b \leftarrow a): 11000100 (把 a 给 b)
13. not(b 取非):01010100
14. and(b&a)结果放在 b 中, 此时 b 一定为 00000000):10110100
15. sub(a-b,即 a-0, 结果为 a):01100001
16. rsr(a 右移):10100000
17. rsl (a 左移): 10100011 (此时 a 还是两个输入之和)
18. rsl (a 左移): 10100011
19. rsr(a 右移):10100000
20. out(a):01000000 (输出 a) 16+8
21. jmp (跳转到最开始): 00110000,00000000 (这是跳转的地址, 要跳转到最开始)
22. halt:10000000

预期输出: output 端口输出 in+in2 的值, output2 端口输出 in-in2 的值
当 in+in2 溢出或者 in+in2=0 时, 执行 halt 指令, 停机

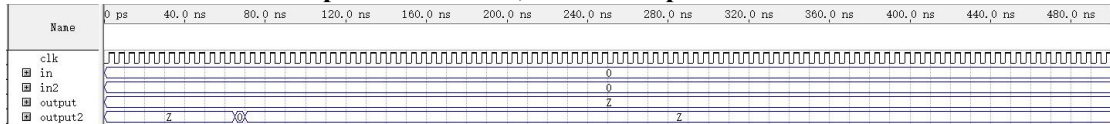
4.3 测试结果



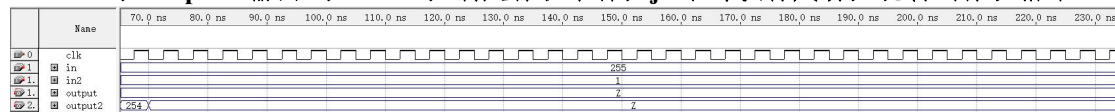
Output2 口输出 100-68=32, output 口输出 100+68=168, 符合预期



0-1=-1 发生溢出, output2 口输出 255.0+1=1, output 口输出 1, 符合预期



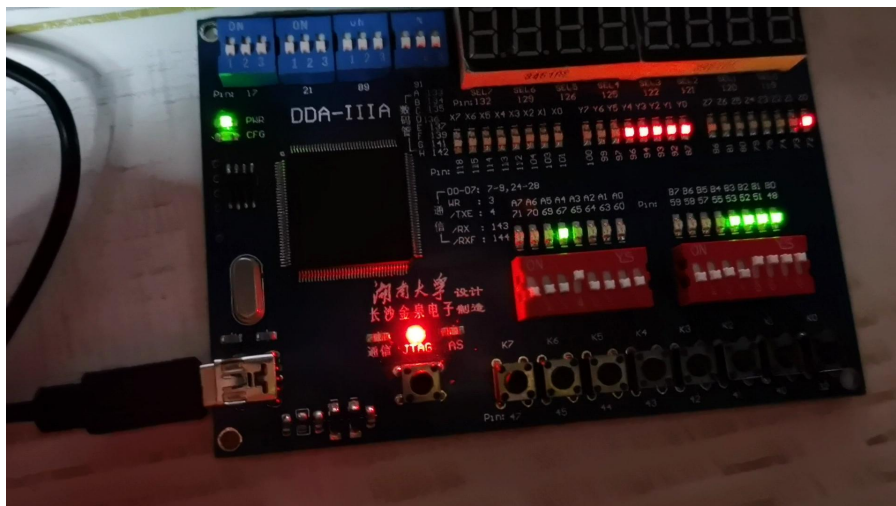
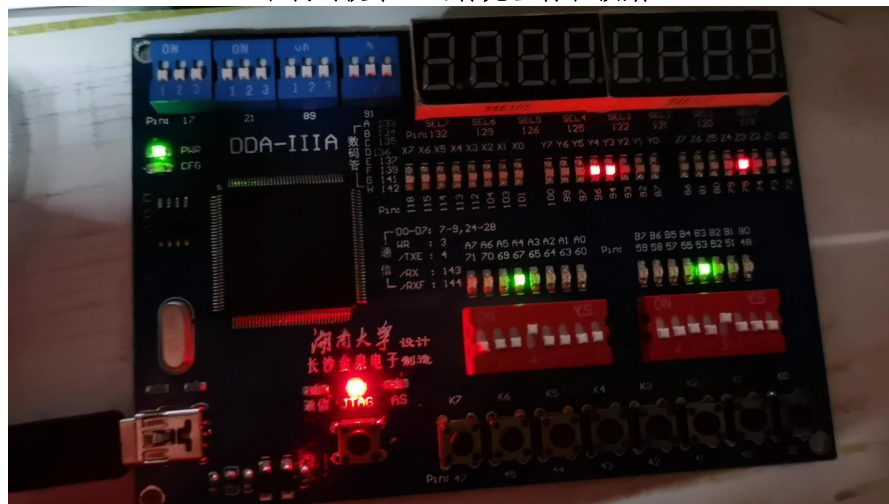
0-0=0, output2 输出 0, 0+0=0, 满足停机条件, jz 语句执行成功, 跳转到停机指令



255-1=254, output2 输出为 254

255+1=256, 发生溢出, 此时 cf=1, jc 语句执行成功, 不再有输出, 跳转到停机指令

下载到板子上运行完全符合预期



4.4 模型机性能分析

模型机能够执行 16 个指令，最终输出正确的结果。且额外实现了两个输入，两个输出的功能，亮点突出。

```

Flow Status                Successful - Sat Jan 01 23:12:57 2022
Quartus II Version         9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name              CPU
Top-level Entity Name      CPU
Family                    Cyclone II
Device                    EP2C5T144C8
Timing Models              Final
Met timing requirements    Yes
Total logic elements       210 / 4,608 ( 5 % )
    Total combinational functions 210 / 4,608 ( 5 % )
    Dedicated logic registers  45 / 4,608 ( < 1 % )
Total registers            45
Total pins                 35 / 89 ( 39 % )
Total virtual pins         0
Total memory bits          2,048 / 119,808 ( 2 % )
Embedded Multiplier 9-bit elements 0 / 26 ( 0 % )
Total PLLs                 0 / 2 ( 0 % )

```

用到了 210 个逻辑门，45 个寄存器，用于实现 CPU 各模块的逻辑运算功能和存储功能
 用到了 35 个 pins，其中 16 个用来实现 2 输入，16 个用来实现 2 输出，1 个用来输入时钟信号，2 个用来显示 cf 和 zf
 用到了 2048 个存储单元，这是 256*8 的 ram 所用到的
 消耗的资源只占芯片的很小一部分

Timing Analyzer Summary					
	Type	Slack	Required Time	Actual Time	From To
1	Worst-case tsu	N/A	None	11.288 ns	in2[2] reg_group:inst8[r]12
2	Worst-case tco	N/A	None	21.448 ns	IR:inst4[ir[0] output[7]
3	Worst-case tpd	N/A	None	17.287 ns	in[7] output[7]
4	Worst-case th	N/A	None	-6.530 ns	in[3] IR:inst4[ir[3]
5	Clock Setup: 'clk'	N/A	None	34.65 MHz (period = 28.862 ns)	reg_group:in... lpm_ram_io:inst13 altram:sram altsyncram:ram_block altsyncram_jf91:auto_generated ram_blo
6	Total number of failed paths				

时序分析不存在异常情况，并且在实现了两输入，两输出的情况下，模型机依然可以运行的频率达 34.65Mhz，处于正常的范围。

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：

完整、连贯地运用《电路与电子学》所学到的数电知识，熟练掌握现代 EDA 工具基本使用方法，为后续课程学习和今后从事相关工作打下良好的基础或做下一些铺垫。

按照给定的数据通路、数据格式和指令系统，使用 EDA 工具设计一台用硬连线逻辑控制的简易计算机。

遇到的困难：

1. 代码中存在个别 bug，仿真时发现异常，即执行 out 的时候总是只输出 a 寄存器的值（即使我给的代码是输出 b 寄存器或者 c 寄存器的值）
2. 想要设计一个具有两输入，两输出的 CPU，但是遇到了非常大的困难

解决的办法：

1. 检查了寄存器的代码，未发现异常；再检查控制信号的代码，发现 out 指令没有被包含在 ALU 的 m 信号中，而实际上是需要被包含在其中的，否则 ALU 会传递源寄存器口的数据给移位逻辑，而不是目的寄存器口的数据，这样就会导致错误。改过来以后仿真波形就正确了。
2. 实现两输入和两输出意味着存在两组输入的 pin 口和两组输出的 pin 口，由于所有的输入输出数据都经过总线，控制信号又不能区分是哪组 pin 口，基于当前存在的模块是没办法实现两输入输出的设计两输入，两输出的功能的关键在于要有一个存储部件记忆当前的输入（或输出）是从哪个 pin 口执行的。因此需要自己额外设计两个小部件。经过自己创新设计，努力钻（折）研（磨）6 个小时后，终于设计出来了。

经验教训：

设计模型机尤其是设计两输入和两输出的时候耗费了太多时间，主要还是因为我太急于求成，想着一下子搞出来，一旦遇到问题就一头往里面钻，钻了两三个小时以后脑袋就不好使了，得不偿失，正确的做法应该是钻了一个小时没钻出来就先放在，慢慢酝酿新的思路。

虽然说这个东西有点小折磨，但是收获也是很多的，首先是自己对模型机的流程十分熟悉了，还能加入自己的一点创意进去，可以说是在理解的基础上有自己的创新。相信对以后的专业课也会有帮助的。

还学会了一个道理，好事多磨，有的看起来很难的东西，经过不懈的努力以后也是可以做好的。但是如果不敢尝试，或者想着摸鱼，就肯定做不好。