模型机设计报告

班级 信安 2302 班 姓名 石云博 学号 202308060227

一、设计目的

本课程力图以“培养学生现代数字系统设计能力”为目标，贯彻以 CPU 设计为核心， 以层次化、模块化设计方法为抓手的组织思路，培养学生设计与实现数字系统的能力。 完整、连贯地运用课程所学知识，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、按照给定的结构框架、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、要求灵活运用各方面知识，使得所设计的计算机具有较佳的性能；

3、对所设计计算机的性能指标进行分析，整理出设计报告。

三、详细设计

3.1、设计的整体架构

数据格式 数据采用 8 位二进制补码表示，其中最高位（第 7 位）为符号位，其数值表示范围为： $-128 \leqslant X \leqslant +127$。

指令格式 除立即数指令是双字节外，其它指令均为单字节，指令高 4 位为操作码，表示指令执行 的操作，用以区分指令；低 4 位为地址码，指明操作数的来源。

寻址方式 寻址方式是确定下一条要执行的指令地址和数据地址的方式，包含指令寻址和数据寻址。

1）指令寻址 指令寻址分为顺序寻址和跳转寻址。 顺序寻址是通过程序计数器 PC 加 1 自动形成下一条指令的地址。 跳转寻址是转移类指令执行时，将转移地址从寄存器 R3 取出，写入程序计数器 PC。

2）数据寻址 指令系统提供灵活的数据寻址方式，用尽量短的地址码提供操作数地址。本模型机共有 3 种数据寻址方式。立即寻址操作数包含在指令中，跟在操作码后面，作为指令的一部分，因此也叫立即数。寄存器直接寻址 操作数存放在通用寄存器中。，指令地址码 2 位一组分别表示两个寄存器编号。此时指 令格式如下图所示，Rs 和 Rd 分别表示两个操作数所在的寄存器编号，其中 Rs 是源寄存器 编号，Rd 是目的寄存器编号。

寄存器间接寻址 操作数存放在存储器中，地址码中有一个寄存器编号为“00”， 此时存放操作数的存储 单元地址在寄存器R0中，通过访问寄存器R0获取存储单元地址，再从存储器对应单元读 取的才是操作数，具体过程为：

指令系统

指令系统有 12 条指令，具体格式见指令系统表。应该指出的是，各条指令的编码形式 可以多种多样。为了叙述方便，下面采用汇编符号对指令进行描述，其中 Rs 和 Rd 分别表示“源”和“目的”寄存器,源和目的寄存器可以是通用寄存器中的任意一个寄存器，M 表示地址在寄存器 R0 中的存储单元。

模型机结构框架

计算机的工作过程可以看作是许多不同的数据流和控制流在各模块之间流动，数据在控 制信号的控制下进行流动，控制信号决定了数据流动的时间和方向。数据流所经过的路径称 作数据通路。数据通路不同，指令执行的操作过程就不同，模型机的结构也就不一样。本模 型机的结构框架如图 1 所示：

模块功能

存储器、指令寄存器、算术单元和通用寄存器之间互连的通道称为总线 BUS，它由 8 根

导线组成，同时传输 8 位数据，实现模块间的数据交互。

程序计数器 PC：存放当前欲执行指令在 RAM 中的存储地址。LD_PC=1 时，PC 装载跳

转地址，IN_PC=1 时，PC 进行自加 1 操作。

3-1 选择器：选择 RAM 的地址来源，其有 3 个输入，每个输入 8 位，分别接程序计数

器 PC 和通用寄存器的 S 口和 D 口，由控制信号 S2、S1 选择其中一个传送至 RAM 的地址

输入端。

存储器 RAM：存放指令和数据，其大小为 256×8 位，即有 256 个存储单元，每个单元

存放 8 位数据，所以该 RAM 有一个 8 位的地址输入端以访问每个存储单元，一个双向的 8

位数据输入输出端。RAM 每个存储单元存放着一条指令或一字节数据。指令在 RAM 中是

顺序存放的，如 0 号存储单元存放第一条指令，1 号存储单元存放第二条指令，依次存放。

从存储器的哪一个单元读取指令，由程序计数器 PC 提供单元地址。指令顺序执行时，即指

令从存储器中按顺序取出来，先取出第一条，再取第二条…，这时程序计数器 PC 自加 1 计

数，指向顺序执行指令在存储器中的单元地址。当执行跳转指令时，程序计数器 PC 中原有

的计数值失效，跳转指令的转移地址从寄存器 R3 送到程序计数器 PC 中，PC 从新的值开始

继续计数。当从存储器中取出数据，注意是取数据，不是取指令，或者向存储器存放运算结

果时，存储单元的地址由通用寄存器提供。

指令寄存器 IR：存放当前执行的指令。

SM：指示当前周期是取指周期还是执行周期。SM=0，表示当前是取指周期，SM=1，

表示当前是执行周期。

指令译码器：解析指令。根据指令寄存器 IR 提供的 8 位指令编码，解析是哪条指令。

控制信号产生逻辑：根据指令译码器解析的指令，产生该指令执行所需的控制信号。

2-1 选择器：选择通用寄存器的数据来源。

通用寄存器：保存操作数和中间计算结果，其有 4 个 8 位寄存器 R0、R1、R2、R3，控制信号 SR1、SR0 选择 4 个寄存器中的一个作为源寄存器，源寄存器的数据从 S 口输出；

DR1、DR0 选择 4 个寄存器中的一个作为目的寄存器，目的寄存器的数据从 D 口输出，WE 是写控制信号，在 DR1、DR0 的配合下，将其输入端的数据写入目的寄存器。

算术单元 AU：实现算术运算，并在执行 MOVA、MOVB、OUT 指令时负责将数据送至总线 BUS。

状态寄存器 PSW：存放 SUB 指令产生的状态位 G。

指令的执行

指令执行与模型机结构、指令执行方式有关。指令可以串行执行，也可以并行执行。本 设计采用串行工作方式，即“读取指令—执行指令—再读取指令—再执行指令……”。串行工 作方式虽然工作速度和效率都要差一些，但它的控制简单。本机一条指令需要两个时钟周期 完成，一个时钟周期读取指令，一个时钟周期执行指令。

读取指令的时间随所使用的 RAM 的性能而异。执行指令的时间依据控制流和数据流所经过的路径与各级门的最大延迟而定。本机中写入 RAM 和通用寄存器的操作显然不能发生在“执行阶段”的任意时刻，它必须是在运算结果已经产生，并被传送到总线的适当时刻才能“写”，这就需要时钟脉冲来控制时序。

1、读取指令的过程

要求完成的操作：从 RAM 中取出指令写入指令寄存器 IR，PC 自加 1，SM 变为 1。

具体过程为：程序计数器 PC 中的地址经 3-1 选择器送至 RAM 的地址输入端；在 RE 和地址的共同作用下,指令在时钟上升沿从 RAM 中读出送至总线 BUS;在 LD_IR 的控制下，BUS 上的指令在时钟下降沿写入指令寄存器 IR；同时程序计数器 PC 自加 1，指向下一条指令在 RAM 中的存放地址；SM 由 0 变为 1，指示下一周期为指令的执行周期。

2、指令的执行过程

每条指令的取指过程都相同，不同的是执行过程。接下将指令划分为数据传送类指令、算术运算类指令、转移类指令、输入输出类指令和停机指令，分别介绍各类指令的执行过程。

每条指令执行完，SM 由 1 变为 0，指示下一周期为读取指令周期。

1）数据传送类指令的执行过程

MOVA　Rd，Rs 要求完成的操作：源寄存器 Rs 中的数据写入目的寄存器 Rd，即（Rs）→Rd。

执行过程：根据控制信号 SR1、SR0 选择源寄存器 Rs 的数据从通用寄存器 S 口输出，在 AC3～AC0 和 AU_EN 的控制下，经 AU 送入总线 BUS；S0 为 1，BUS 上的数据传送至通用寄存器的输入端；在 WE 和 DR1、DR0 的控制下，时钟下降沿将输入端的数据写入目的寄存器 Rd。

MOVB　M，RS 要求完成的操作：源寄存器 Rs 中的数据写入 RAM 的某个存储单元，该单元的地址存放在寄存器 R0 中，即（RS）→（R0）。

执行过程：控制信号 DR1、DR0 为 00（寄存器 R0 的编号），从通用寄存器 D 口输出 R0 中的内容，控制信号 S2、S1 为 10，R0 的内容通过 3-1 选择器到达存储器 RAM 的地址输入端；控制信号 SR1、SR0 选择源寄存器 Rs 的数据从通用寄存器 S 口输出，在 AC3～AC0 和 AU_EN 的控制下，经 AU 送入总线 BUS，在 WR 的控制下，时钟上升沿将 BUS 上的数据写入存储器 RAM。

MOVC Rd，M

要求完成的操作：寄存器 R0 给出 RAM 的单元地址，读取该单元的数据写入目的寄存器 Rd，即（(R0)）→Rd。

执行过程：控制信号 SR1、SR0 为 00（寄存器 R0 的编号），从通用寄存器 S 口输出 R0 中的内容，控制信号 S2、S1 为 01，R0 的内容通过 3-1 选择器到达存储器 RAM 的地址输入端；在 RE 控制下，时钟上升沿从 RAM 中读取数据，送入总线 BUS；S0 为 1，BUS 上的数据传送至通用寄存器的输入端；在 WE 和 DR1、DR0 的控制下，时钟下降沿将输入端的数据写入目的寄存器 Rd。

MOVD R3，PC 要求完成的操作：程序计数器 PC 中的内容写入寄存器 R3，即（PC）→R3。

执行过程：控制信号 S0 为 0，PC 中的内容传送至通用寄存器的输入端；WE 为 1，DR1、DR0 为 11（寄存器 R3 的编号），时钟下降沿将 PC 的内容写入寄存器 R3。

MOVI IMM 这条指令是双字节指令，第一字节为指令码，第二字节为立即数。

要求完成的操作：将指令中的立即数写入寄存器 R0，即 IMM→R0。

执行过程：控制信号 S2、S1 为 00，程序计数器 PC 中的地址通过 3-1 选择器传至存储器 RAM 的地址输入端，在 RE 控制下，时钟上升沿将立即数从 RAM 中读出并送入总线 BUS；S0 为 1，BUS 上的数据传送至通用寄存器的输入端；WE 为 1，DR1、DR0 为 00（寄存器 R0 的编号），时钟下降沿将数据写入寄存器 R0。在 IN_PC 控制下，时钟下降沿 PC 加 1 计数，指向下一条指令在 RAM 中的存放地址。

2）算术运算类指令的执行过程

ADD

Rd，Rs 要求完成的操作：源寄存器 Rs 和目的寄存器 Rd 的数据相加，和写入目的寄存器 Rd，即（Rs）+（Rd）→Rd。

执行过程：控制信号 SR1、SR0 选择源寄存器 Rs 的数据从 S 口输出，控制信号 DR1、DR0 选择目的寄存器 Rd 的数据从 D 口输出；在 AC3～AC0 和 AU_EN 的控制下，在 AU 中进行加法运算后将相加的和送入总线 BUS；S0 为 1，BUS 上的数据传送至通用寄存器的输入端；在 WE 和 DR1、DR0 的控制下，时钟下降沿将输入端的数据写入目的寄存器 Rd。

SUB　Rd，Rs 要求完成的操作：寄存器 Rd 的数据减去 Rs 的数据，差写入寄存器 Rd，即（Rd）-（Rs）→Rd。 执行过程：控制信号 SR1、SR0 选择源寄存器 Rs 的数据从 S 口输出，控制信号 DR1、DR0 选择目的寄存器 Rd 的数据从 D 口输出；在 AC3～AC0 和 AU_EN 的控制下，在 AU 中进行减法运算后将相减的差送入总线 BUS；S0 为 1，BUS 上的数据传送至通用寄存器的输入端；在 WE 和 DR1、DR0 的控制下，时钟下降沿将输入端的数据写入目的寄存器 Rd。SUB 指令影响状态位 G，如果 Rd>Rs，则 G=1，否则 G=0。

3）转移类指令的执行过程

JMP 要求完成的操作：寄存器 R3 的内容写入程序计数器 PC，即（R3）→PC。

执行过程：控制信号 SR1、SR0 为 11（寄存器 R3 的编号），从通用寄存器 S 口输出 R3 中的内容，在 LD_PC 的控制下，时钟下降沿将 R3 的内容写入程序计数器 PC。

JG 要求完成的操作：仅当 G=1，将寄存器 R3 的内容写入程序计数器 PC，否则 PC 的内容保持不变，即 IF(Rd>Rs)，THEN G=1，ELSE G=0。

执行过程：控制信号 SR1、SR0 为 11（寄存器 R3 的编号），从通用寄存器 S 口输出 R3 中的内容，如果条件满足（即 G=1），在 LD_PC 控制下，时钟下降沿将 R3 的内容写入程序计数器 PC，否则 PC 的内容保持不变。

4）输入输出指令的执行过程

IN　Rd

要求完成的操作：将外设输入的数据写入寄存器 Rd。

执行过程：在 IN_EN 控制下，外设输入的数据送至总线 BUS；控制信号 S0 为 1，BUS 上的数据传送至通用寄存器的输入端；在 WE 和 DR1、DR0 的控制下，时钟下降沿将输入端的数据写入目的寄存器 Rd。

OUT　Rs

要求完成的操作：寄存器 Rs 中的数据输出至外部设备。

执行过程：控制信号 SR1、SR0 选择源寄存器 Rs 的数据从 S 口输出，在 AC3～AC0 和 AU_EN 的控制下，经 AU 送入总线 BUS，在 OUT_EN 控制下将 BUS 上的数据输出至外部设备。

5）停机指令的执行过程

HALT 停机指令，执行完这条指令，模型机进入停机状态，指令 HALT 后面即使还有其他指令，模型机也不再执行。

3.2 各模块的具体实现

整体架构：

指令译码器：

模块代码：

测试结果：

功能仿真：

时序仿真：

结果分析与结论：

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 en 为 0 时，不管 ir 为何值，12 个输出全为 0

当 en 为 1 时：

当 ir=0100….时，mova 输出为 1；

当 ir=010100..时，movb 输出为 1；

当 ir=0110..时，movc 输出为 1；

当 ir=011111..时，movd 输出为 1；

当 ir=1000….时，add 输出为 1；
当 ir=1001….时，sub 输出为 1；
当 ir=1010….时，jmp 输出为 1；
当 ir=1011….时，jg 输出为 1；
当 ir=1100….时，in1 输出为 1；
当 ir=1101….时，out1 输出为 1；
当 ir=111000..时，movi 输出为 1；
当 ir=11110000 时，halt 输出为 1；

AU：

源代码：

资源消耗：

功能仿真：

时序仿真：

结果分析与结论：

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。

当控制信号 au_en 为 1，s 为 1000 时，执行 t=a+b

当控制信号 au_en 为 1，s 为 1001 时，执行 t=b-a

当控制信号 au_en 为 1，s 为 0100 时，执行 t=a

当控制信号 au_en 为 1，s 为 0101 时，执行 t=a

当控制信号 au_en 为 0，s 为 1101 时，执行 t=a

当控制信号 au_en 为 0，s 上述其他时，执行 t=zzzzzzzz

有进位和借位时 gf 为 1，否则为 0；

8 重 3-1 多路复用器

源代码：

资源消耗：

功能仿真：

时序仿真：

结果分析：

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。

s=00 时，控制输出 y 等于 a，正确；

s=01 时，控制输出 y 等于 b，正确；

s=10 时，控制输出 y 等于 c，正确；

s=11 时，控制输出高阻态，正确；

8 重 2-1 多路复用器：

源代码：

资源消耗：

功能仿真：

时序仿真：

分析与结论：

：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 s=0 时，输出 t=a;

当 s=1 时，输出 y=b；

控制信号产生逻辑：

原代码：

资源消耗：

功能仿真：

时序仿真：

结果分析：

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. 指令码 ir 为 10100111 时，au_ac=1010，reg_dr=01，reg_sr=11；
2. mova 指令执行时，sm_en 输出为 1，au_en 输出为 1，mux_s 输出为 1，reg_we 输出为 1；
3. movb 指令执行时，sm_en 输出为 1，ram_wr 输出为 1，au_en 输出为 1，mux_s 输出为 1；
4. movc 指令执行时，sm_en 输出为 1，ram_re 输出为 1，reg_we 输出为 1；
5. movd 指令执行时，sm_en 输出为 1，reg_we 输出为 1；
6. add 指令执行时，sm_en 输出为 1，au_en 输出为 1，mux_s 输出为 1，reg_we 输出为 1；

7. sub 指令执行时，sm_en 输出为 1，au_en 输出为 1，gf_en 输出为 1，mux_s 输出为 1，reg_we 输出为 1；
8. jmp 指令执行时，sm_en 输出为 1,pc_ld 输出为 1；
9. sm 指令为 1 时，表示为指令执行阶段，指令可以正常执行；sm 指令为 0 时表示取指阶段，此时 in_pc、ram_re、id_ir、sm_en 输出为 1；
10. jg 指令执行时，sm_en 输出为 1，au_en 输出为 1；
11. 当 g 与 jg 指令共同执行时，pc_ld 输出为 1；
12. 当 in1 指令执行时，sm_en 输出为 1，in_en 输出为 1，mux_s 输出为 1，reg_we 输出为 1；
13. 当 out1 指令执行时，sm_en 输出为 1，au_en、out_en 输出为 1，mux_s 输出为 1，reg_we 输出为 1；
14. 当 movi 指令执行时，sm_en 输出为 1，pc_in 输出为 1，ram_re 输出为 1，mux_s 输出为 1，reg_we 输出为 1;
15. 当 halt 指令执行时，sm_en 输出为 0，sm 不反转，无法进行下一轮取指操作，指令机停机。

Sm：

原代码：

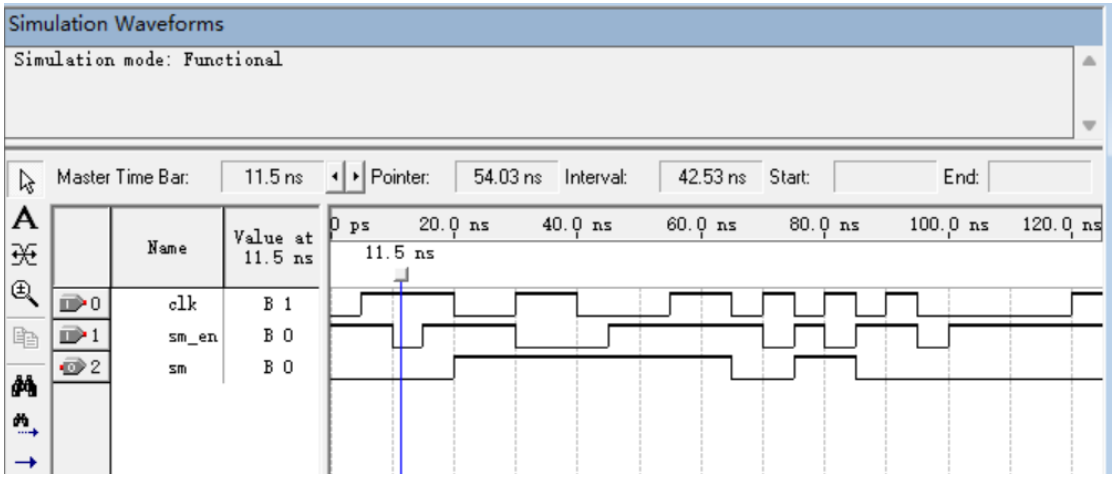
```
module sm(
    input clk,sm_en,
    output reg sm);

initial sm=1'b0;

always @(negedge clk)
begin
    if(sm_en==1'b1) sm<=~sm;
    else sm<=sm;
end
endmodule
```

资源消耗：

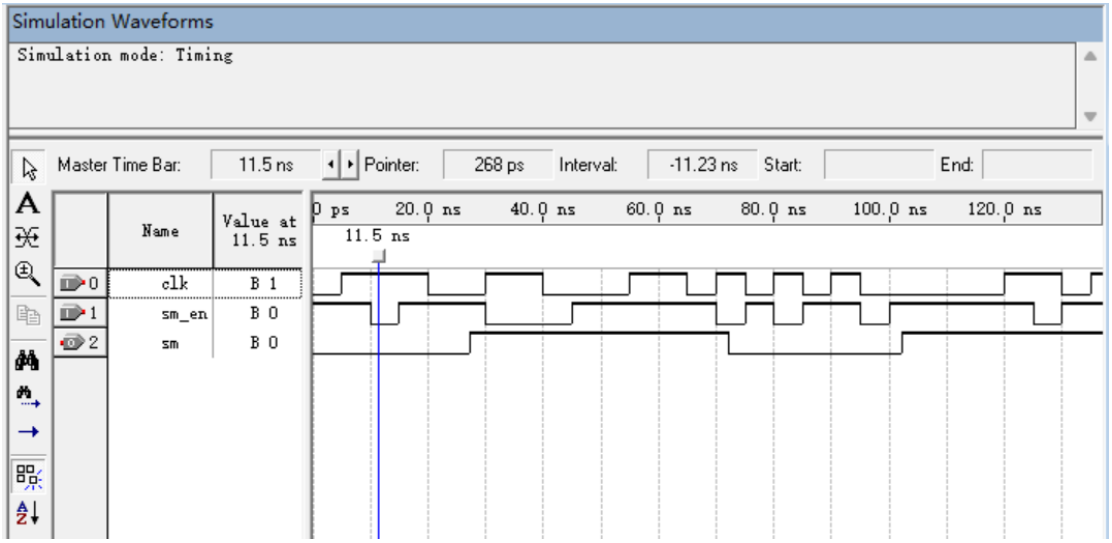
Flow Summary

Flow Status	Successful - Mon Dec 09 21:50:42 2024
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	sm
Top-level Entity Name	sm
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1 / 4,608 (< 1 %)
Total combinational functions	1 / 4,608 (< 1 %)
Dedicated logic registers	1 / 4,608 (< 1 %)
Total registers	1
Total pins	3 / 89 (3 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

功能仿真波形：



时序仿真波形：



结果分析及结论

分析：对于功能仿真，可以看到当 sm_en 信号为 1 有效时,输出 sm 在时钟的下降沿发生翻转，符合功能设计。当 sm_en 信号为 0 时，输出信号 sm 保持不变，正确。

对于时序仿真，其输出结果和功能仿真类似，但存在 7ns 左右的延迟

结论：元件设计符合设计要求，元件内部存在 7ns 左右的延迟

指令寄存器 IR：

源代码：

ir.v

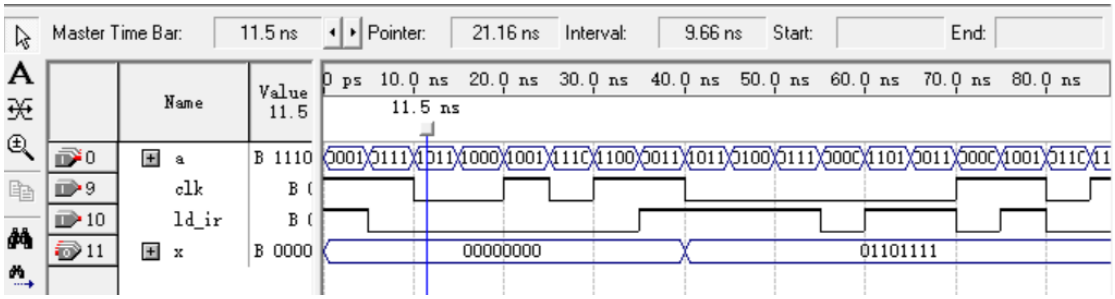
```
1  module ir (
2      input clk,ld_ir,
3      input [7:0] a,
4      output reg [7:0] x=8'b00000000);
5
6  always @(negedge clk)
7  begin
8      if(ld_ir==1'b1) x<=a;
9  end
10 endmodule
```

资源消耗：

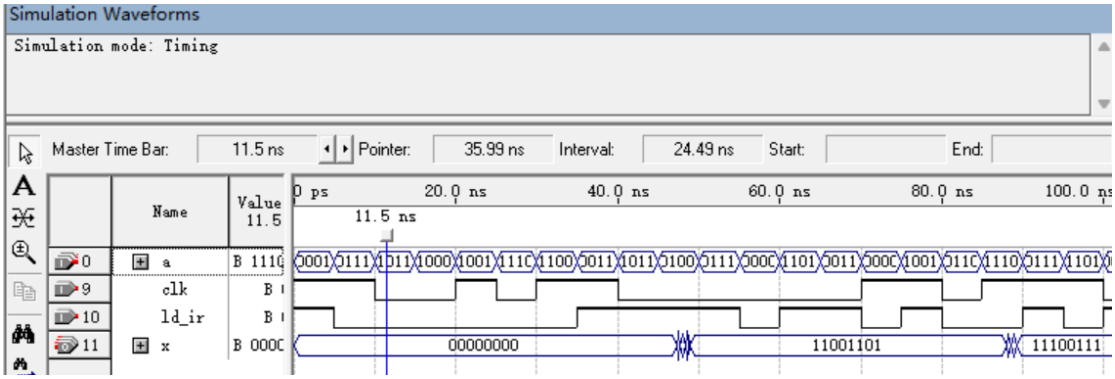
Flow Summary

Flow Status	Successful - Mon Dec 09 21:54:46 2024
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	ir
Top-level Entity Name	ir
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	8 / 4,608 (< 1 %)
Total combinational functions	0 / 4,608 (0 %)
Dedicated logic registers	8 / 4,608 (< 1 %)
Total registers	8
Total pins	18 / 89 (20 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

功能仿真波形



时序仿真波形



结果分析及结论

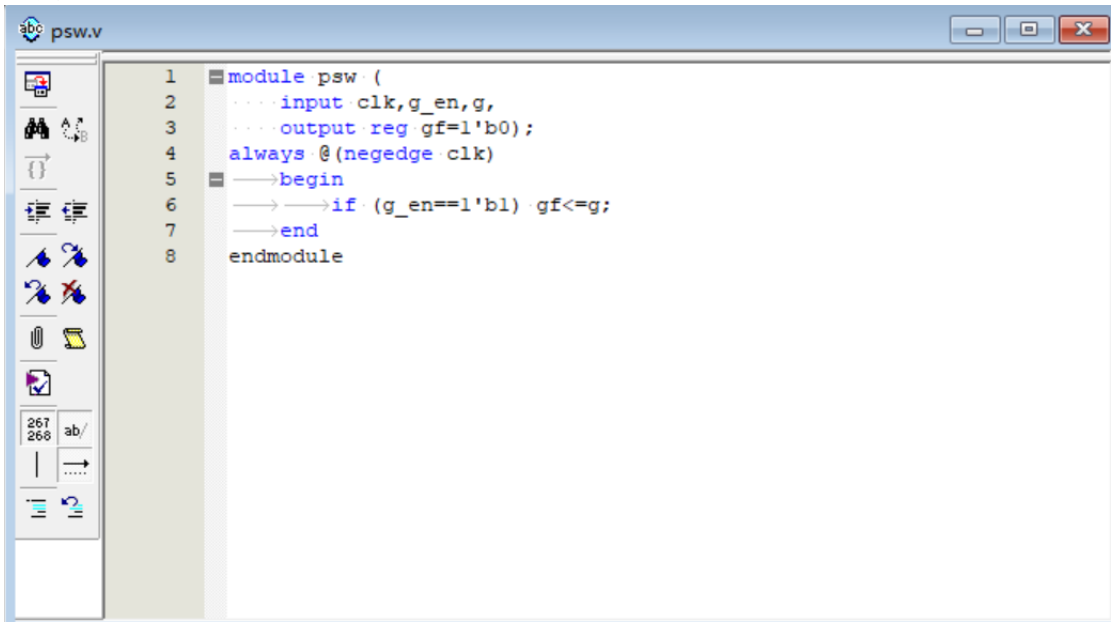
分析：对于功能仿真，在 0-5ns，ld_ir 为 1，在时钟下降沿将输入写入输出，当 ld_ir 为 0 时，输出保持不变，正确

对于时序仿真，可以看到输出存在 9ns 左右的延迟，同时部分时刻输入的变化导致冒险出现，使得输出错误，输出的变化情况大致与功能仿真相同

结论：元件设计符合设计要求，元件内部存在 9ns 左右的延迟

状态寄存器 PSW

源代码

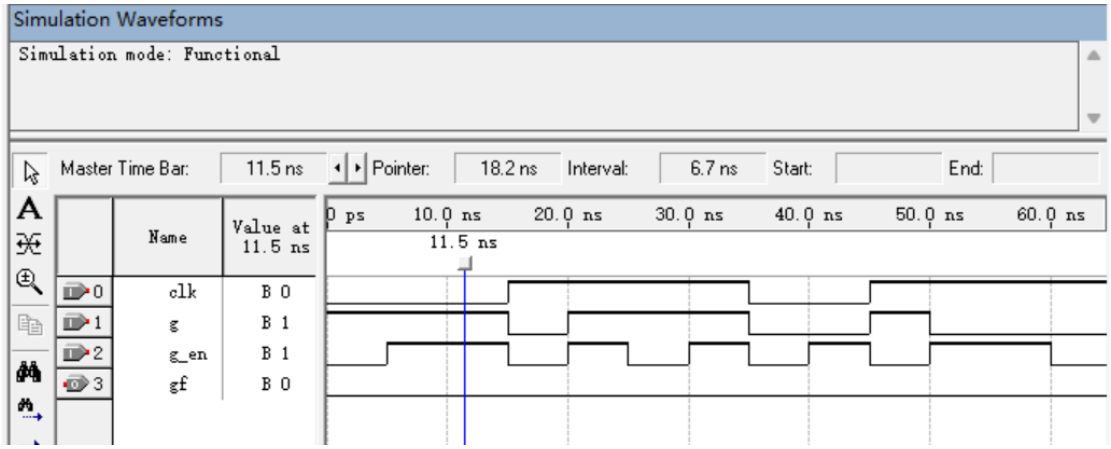


资源消耗：

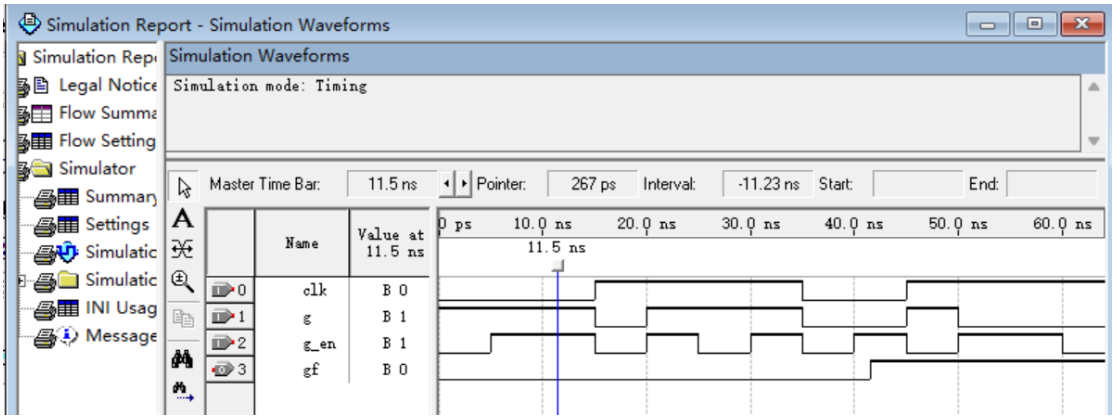
Flow Summary

Flow Status	Successful - Mon Dec 09 21:58:25 2024
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	psw
Top-level Entity Name	psw
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1 / 4,608 (< 1 %)
Total combinational functions	1 / 4,608 (< 1 %)
Dedicated logic registers	1 / 4,608 (< 1 %)
Total registers	1
Total pins	4 / 89 (4 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

功能仿真波形



时序仿真波形



结果分析及结论

分析：对于功能仿真，42-60ns，g_en 为 1，在时钟下降沿，将 g 的值写入输出 gf 中，15-20ns，

g_en 为 0，输出 gf 保持不变，正确

结论：元件设计符合要求，输出 gf 有 7ns 延迟

指令计数器 PC

源代码

pc.v

```
1  module pc (
2      input clk,ld_pc,in_pc,
3      input [7:0] a,
4      output reg[7:0] c=8'b00000000);
5
6   always @(negedge clk)
7  begin
8      if(in_pc==1'b1&&ld_pc==1'b0) c<=c+8'b0000001;
9      else if(in_pc==1'b0&&ld_pc==1'b1) c<=a;
10  end
11  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）

确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功，保存文件。

无错误。

警告信息：

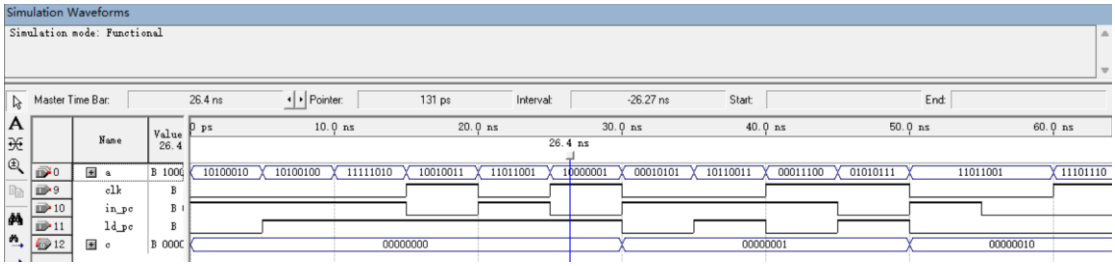
```
Warning: Feature LogicLock is not available with your current license
Warning: No exact pin location assignment(s) for 19 pins of 19 total pins
Warning: Found 8 output pins without output pin load capacitance assignment
Warning: The Reserve All Unused Pins setting has not been specified, and will default to 'As output driving ground'
Warning: Found pins functioning as undefined clocks and/or memory enables
```

资源消耗：

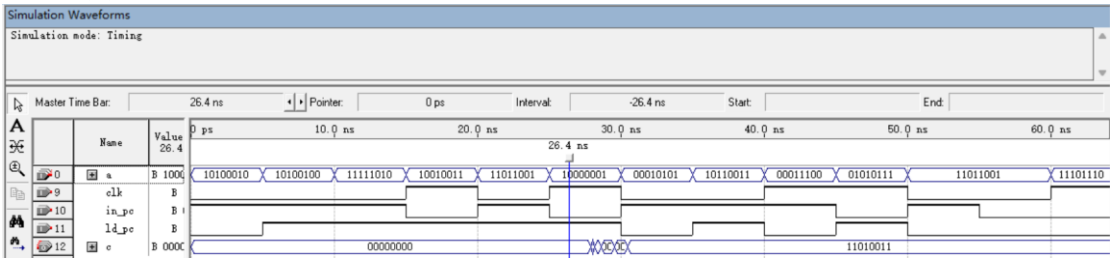
Flow Summary

Flow Status	Successful - Mon Dec 09 22:03:48 2024
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	pc
Top-level Entity Name	pc
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	10 / 4,608 (< 1 %)
Total combinational functions	10 / 4,608 (< 1 %)
Dedicated logic registers	8 / 4,608 (< 1 %)
Total registers	8
Total pins	19 / 89 (21 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

功能仿真波形



时序仿真波形



结果分析及结论

分析：对于功能仿真，0-5ns，in_pc 为 1，ld_pc 为 0，执行地址加 1 操作，15ns-25ns,in_pc 为 0，ld_pc 为 1，执行写入操作，将输入写入到输出中，25-40ns，in_pc 为 0，ld_pc 为 0，数据保持不变，正确

对于时序仿真，存在 9ns 左右的延迟，输出结果大致与功能仿真相同

结论：元件设计符合要求，元件存在 9ns 左右的延迟

通用寄存器组

源代码：

reg_group.v

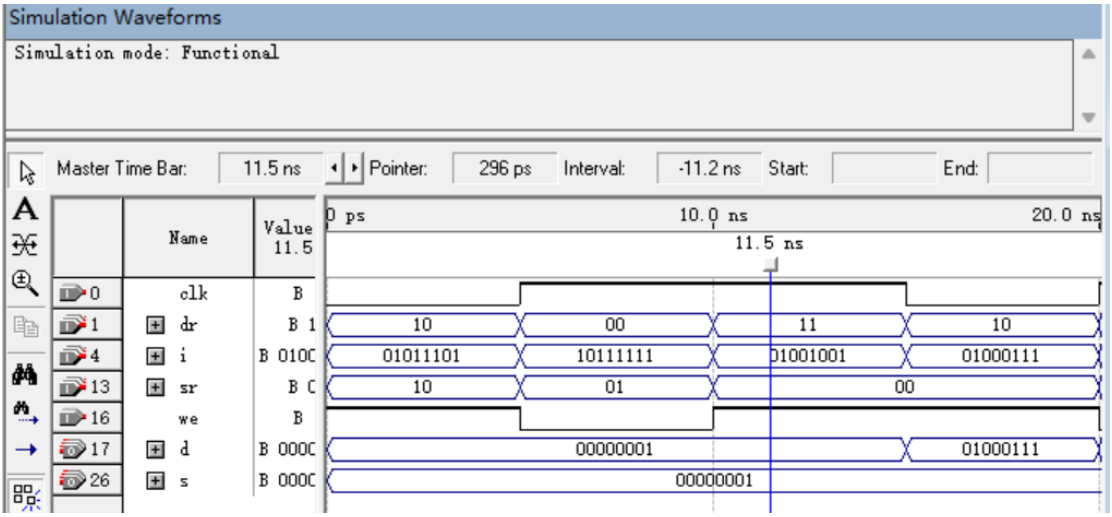
```
1  module reg_group(
2      input clk,we,
3      input [1:0] sr,dr,
4      input [7:0] i,
5      output reg[7:0] s,d);
6    reg[7:0] R0=8'b00000001,R1=8'b00000001,R2=8'b00000001,R3=8'b00000001;
7
8    always @(sr,dr)
9  begin
10     case (sr)
11     2'b00: s=R0;
12     2'b01: s=R1;
13     2'b10: s=R2;
14     default: s=R3;
15     endcase
16
17     case (dr)
18     2'b00: d=R0;
19     2'b01: d=R1;
20     2'b10: d=R2;
21     default: d=R3;
22     endcase
23   end
24
25   always @(negedge clk)
26 begin
27     if(we==1'b1)
28     begin
29         case (dr)
30         2'b00: R0<=i;
31         2'b01: R1<=i;
32         2'b10: R2<=i;
33         2'b11: R3<=i;
```

资源消耗：

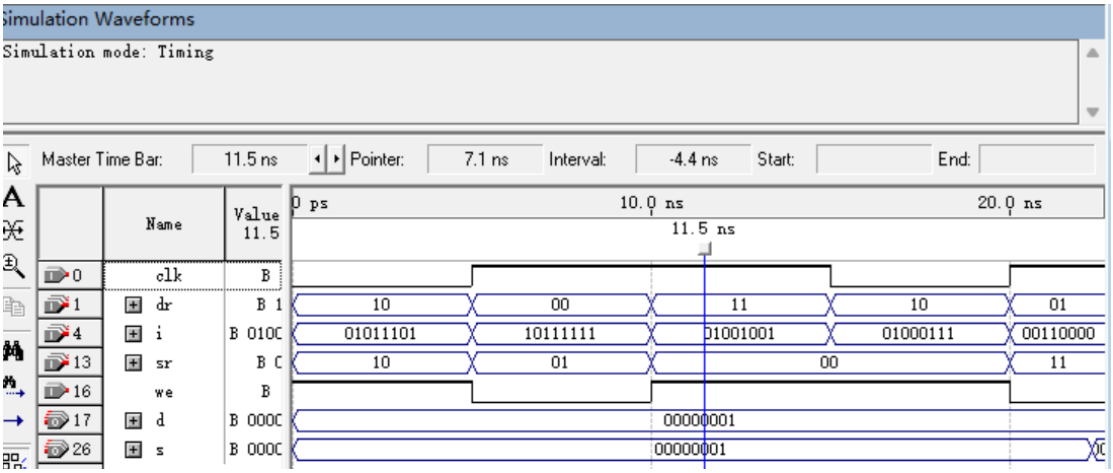
Flow Summary

Flow Status	Successful - Mon Dec 09 22:09:22 2024
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	reg_group
Top-level Entity Name	reg_group
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	40 / 4,608 (< 1 %)
Total combinational functions	40 / 4,608 (< 1 %)
Dedicated logic registers	32 / 4,608 (< 1 %)
Total registers	32
Total pins	30 / 89 (34 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

功能仿真波形：



时序仿真波形



结果分析及结论

分析：对于功能仿真，当 we=1 时进行写入操作，在时钟信号的下降沿根据 dr 的值将 i 写入寄存器（R0、R2、R1、R0，在波形中未体现），当 dr=00 时将 R0 的值从 d 中输出，dr=01 时将 R1 的值从 d 中输出，dr=10 时将 R2 的值从 d 中输出，dr=11 时将 R3 的值从 d 中输出。当 sr=00 时将 R0 的值从 s 中输出，sr=01 时将 R1 的值从 s 中输出，sr=10 时将 R2 的值从 s 中输出，sr=11 时将 R3 的值从 s 中输出。对于时序仿真，基本功能与波形和功能仿真类似，但输出存在 10s 左右的延迟，同时其中较多变化出现冒险。

四、系统测试

4.1 测试环境：

开发软件:QuartusII 9.1 Build 35003/24/2010 SP 2 SJ Full Version 操作系统: Windows 11 家庭中文版 FPGA 学习板芯片信号:Cyclone II EP2C5T144C8

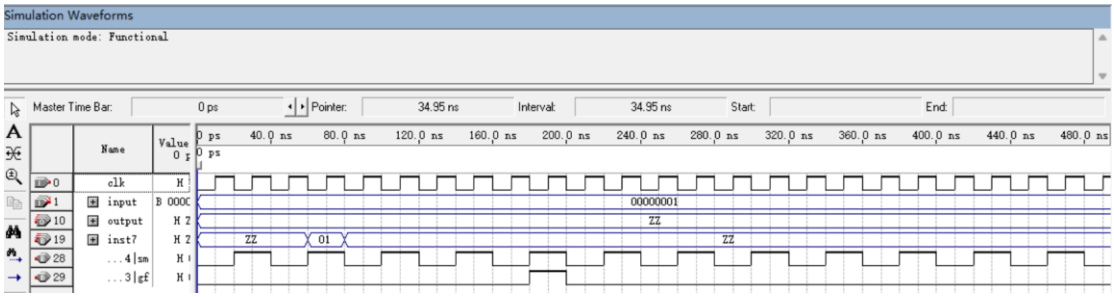
4.2 测试代码：

RAM 地址	指令	指令二进制编码	指令十六制编码	
0，1:	JMP 04H	(00110000 00000100)	(30H 04H)	
4:	IN A	(00100000)	(20H)	A=10000011 (83H)
			（注：A 中的 83H 是由外部输入的）	
5:	OUT A	(01000000)	(40H)	
6:	MOV M A	(11001100)	(CCH)	C=10000000 (80H)
		（注：C 中的 80H 是在通用寄存器组设计时对 C 初始化的值）		
7:	MOV B M	(11000111)	(C7H)	B=10000011 (83H)
8:	SUB A,B	(01100001)	(61H)	A=00000000 (00H), Z=1
9,10:	JZ 10H	(00110001 00010000)	(31H 10H)	
16:	NOT A	(01010000)	(50H)	A=11111111 (FFH)
17:	AND A,C	(10110010)	(B2H)	A=10000000 (80H)
18:	ADD A,B	(10010001)	(91H)	A=00000011 (03H),C=1
19,20:	JC 18H	(00110010 00011000)	(32H 18H)	
24:	NOP	(01110000)	(70H)	
25:	RSR A	(10100000)	(A0H)	A=10000001 (81)
26:	OUT A	(01000000)	(40H)	
27:	RSL B	(10100111)	(A7H)	B=00000111 (07H)
28:	MOV A,B	(11000001)	(C1H)	A=00000111 (07H)
29:	OUT A	(01000000)	(40H)	
30:	HALT	(10000000)	(80H)	
31:	ADD A,B	(10010001)	(91H)	
32:	OUT A	(01000000)	(40H)	

4.3 测试结果：

small_computer.mif

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	10100011	00010001	00000000	00000000	00000000	00000000	00000000	11000100
8	01001001	01100100	10010110	01111100	10001101	11100000	00000010	10110011
16	11110000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
24	00000000	00000000	00000000	00000000	01010011	01100100	11100000	00001001
32	10010001	10110011	11010000	11100000	00000111	01001100	11100000	00000001
40	10100011	00000000	00000000	00000000	00000000	00000000	00000000	00000000
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
64	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
72	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
80	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
88	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
96	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
104	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
112	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
120	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
128	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000



4.4 性能分析：

Flow Status	Successful - Sat Dec 21 22:59:25 2024
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	small_computer
Top-level Entity Name	small_computer
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	159 / 4,608 (3 %)
Total combinational functions	159 / 4,608 (3 %)
Dedicated logic registers	50 / 4,608 (1 %)
Total registers	50
Total pins	17 / 89 (19 %)
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 %)

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock
1	Worst-case tsu	N/A	None	10.342 ns	input[2]	reg_group:inst12|r1[2]	--
2	Worst-case tco	N/A	None	16.416 ns	ir:inst|x[1]	output[4]	clk
3	Worst-case tpd	N/A	None	14.648 ns	input[2]	output[2]	--
4	Worst-case th	N/A	None	-6.348 ns	input[4]	reg_group:inst12|r3[4]	--
5	Clock Setup: 'clk'	N/A	None	45.99 MHz (period = 21.744 ns)	reg_group:inst12|r0[2]	lpm_ram_io:inst1|altram:sram|altsyncram:ram_block|altsyncram_9ma1:auto_generated|ram_block1a0~porta_datain_reg3	clk
6	Total number of failed paths						

分析：此程序占用资源数为 159，时钟频率为 45.99MHz，性能比较良好，占用资 源也比较少，并且设计出的程序符合实验要求。

五、实验总结、必得体会及建议

需要掌握的理论：理解模型机中数据的格式，在该模型及中数据采用 8 位二 进制定点补码表示，其中最高位数位符号位。其指令共有 8 位，前 4 位为操作码， 后四位分别表示目的寄存器和源寄存器。而对于该模型机的指令系统共有 16 条， 在上述分析中已经阐述。同时我们应该了解简易模型机的内部细分结构和工作原 理。同时熟悉多路复用器、移位逻辑和控制信号产生逻辑 sm,ir,psw,pc,reg_等器件的工作原理。具体来说就是学会其功能和指令 译码器的汇编符号到编码，再到最后的功能。了解其具体元器件的引脚关系并且 学会用 Verilog 语言在 quartus 上进行代码的编写。 遇到的困难：一开始很无从下手，只是把各个部分的代码完成，但是并不会 组装。一开始把所有的连线都一一连在对应位置，不过这样看上去非常冗余不美 观。后来学会了用标号来阐述输入输出引脚，这样看上去不仅清晰美观，而且可 读性强。到运行 bdf 文件的时候各种顶层文件、路径名称以及引脚的报错。折腾 好这些文件后开始调试代码，而每次的波形图总是不能完全正确地展示每个输 出。在重

复调试了每个模块后，终于能够运行不报错。不过并不清楚该如何减少 空间占用内存和耗时。通过不断重写代码，能够有效地减小耗时和占用内存。不 过到后边内存和耗时似乎不能同时兼得。 解决办法：在一开始不太会用 quartus 的波形仿真，通过上网搜教程并且 询问老师和同学得以解决此问题。最头大的是不会分析时序仿真中出现的那么多 冒险，通过广泛的在网上查阅资料，开始不断修改代码来减少冲突冒险的产生。同时通过凌老师的录课找到了连接模型机的一些启发，可以顺着老师的思路去完 成整个机器的连接。

5.2 对本实验内容、过程和方法的改进建议（可选项）。

希望实验指导书上可以不光有理想的输入输出关系，还可以有一些实际中会 出现的问题如冲突的出现，以及对这一问题的分析和解决方法。