实验二 模型机组合部件的实现（一）

班级 信安 2101 班 姓名 孙照海 学号 202109070105

一、实验目的

1. 了解简易模型机的内部结构和工作原理。
2. 熟悉译码器、运算器的工作原理。
3. 分析模型机的功能，设计指令译码器。
4. 分析模型机的功能，设计 ALU。

二、实验内容

1. 用 VERILOG 语言设计指令译码器；
2. 用 VERILOG 语言设计 ALU。

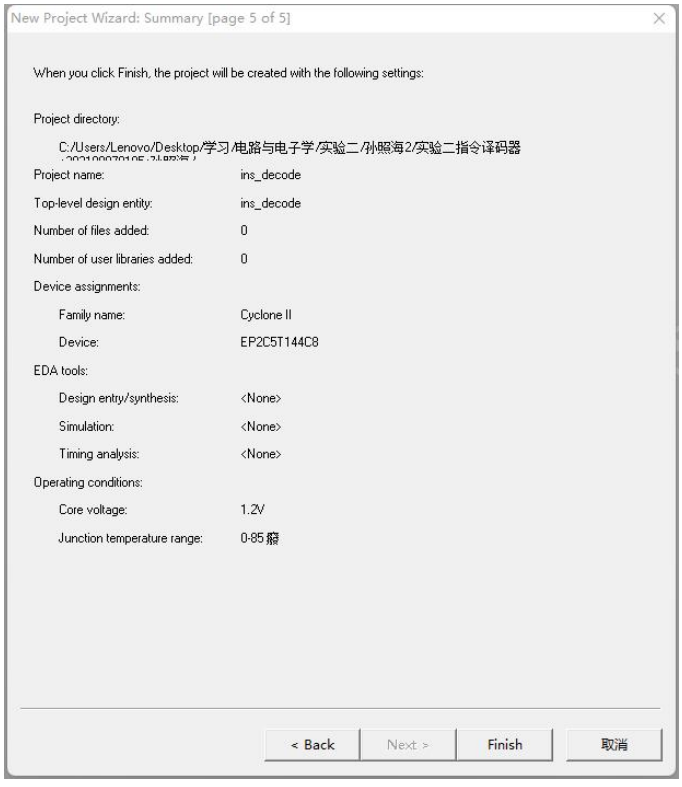
三、实验过程

1、指令译码器

A）创建工程（选择的芯片为 family=Cyclone II；name=EP2C5T144C8）

步骤：左上角 file->New Project Wizard->选择工程位置和工程名->选择芯片 Cyclone II，available device 中选择 EP2C5T144C8->点击 next->最后点击 finish 完成创建工程

工程创建图：



B）编写源代码

根据实验指导和要求实现的功能写出对应的 Verilog 代码。

步骤：左上角 file->new->Verilog hdl file->编写代码（模块名需与工程名一致）->编译成功后保存到工程文件中。

代码截图：

```
module ins_decode(en,ir,mova,movb,movc,add,sub,andl,notl,rsr,rsl,jmp,jz,jc,inl,outl,nop,halt);
  input wire en;
  input wire [7:0] ir;
  output reg mova,movb,movc,add,sub,andl,notl,rsr,rsl,jmp,jz,jc,inl,outl,nop,halt;
  always @(ir) //ir不为0时
    begin {mova,movb,movc,add,sub,andl,notl,rsr,rsl,jmp,jz,jc,inl,outl,nop,halt}=0;
    if(en==1'b1)
      begin
      if(ir[7:4]==4'b1100)
      begin
        if(ir[3:2]==2'b11) movb=1;
        else if(ir[1:0]==2'b11) movc=1;
        else mova=1;
      end
      else if(ir[7:4]==4'b1001) add=1;
      else if(ir[7:4]==4'b0110) sub=1;
      else if(ir[7:4]==4'b1011) andl=1;
      else if(ir[7:4]==4'b0101) notl=1;
      else if(ir[7:4]==4'b1010)
      begin
        if(ir[1:0]==2'b00) rsr=1;
        else if(ir[1:0]==2'b11) rsl=1;
      end
      else if(ir[7:2]==6'b001100)
      begin
        if(ir[1:0]==2'b00) jmp=1;
        else if(ir[1:0]==2'b01) jz=1;
        else if(ir[1:0]==2'b10) jc=1;
      end
      else if(ir[7:4]==4'b0010) inl=1;
      else if(ir[7:4]==4'b0100) outl=1;
      else if(ir[7:0]==8'b01110000) nop=1;
      else if(ir[7:0]==8'b10000000) halt=1;
      end
    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）

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。

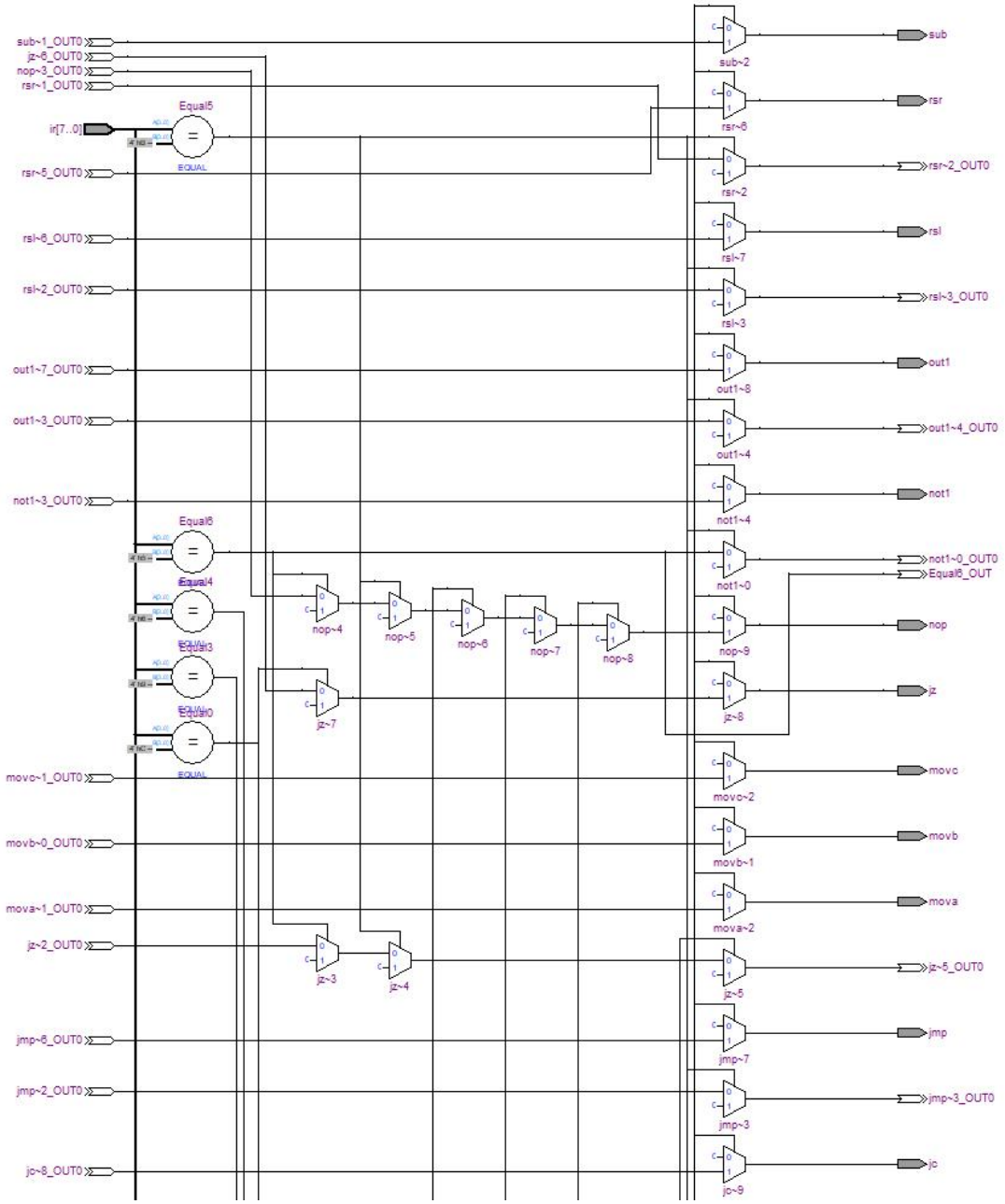
警告信息：

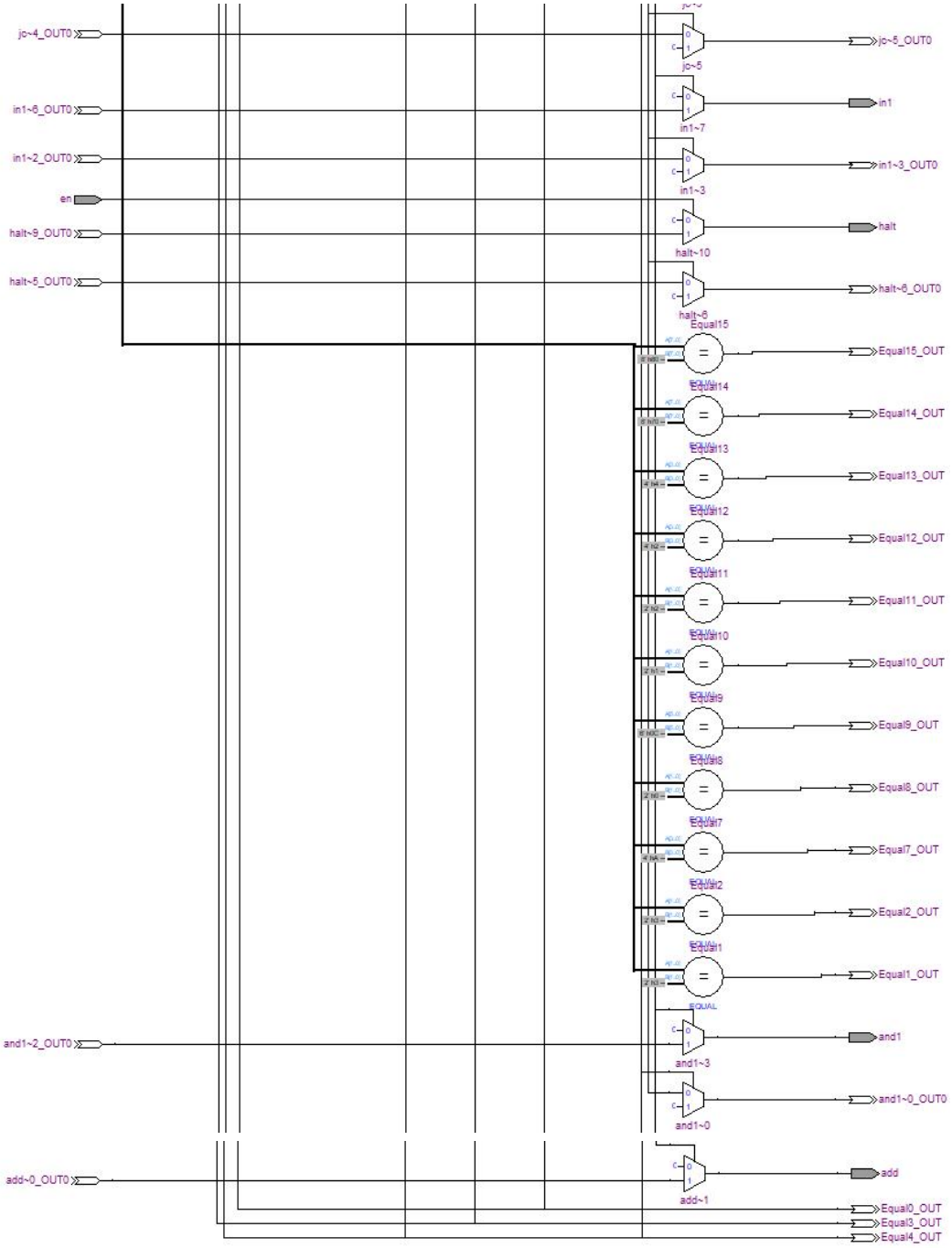
```
Warning (10235): Verilog HDL Always Construct warning at ins_decode.v(7): variable "en" is read inside the Always Construct but isn't in the Always Construct's Event Control
Warning: Feature LogicLock is not available with your current license
Warning: No exact pin location assignment(s) for 25 pins of 25 total pins
Warning: Some pins have incomplete I/O assignments. Refer to the I/O Assignment Warnings report for details
Warning: Found 16 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 - Wed Nov 16 22:11:03 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	ins_decode
Top-level Entity Name	ins_decode
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	30 / 4,608 (< 1 %)
Total combinational functions	30 / 4,608 (< 1 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	25 / 89 (28 %)
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


sub~1_OUT0
jz~6_OUT0
nop~3_OUT0
rsr~1_OUT0
Equal5
ir[7..0]
EQUAL
rsr~5_OUT0
rsl~6_OUT0
rsl~2_OUT0
out1~7_OUT0
out1~3_OUT0
not1~3_OUT0
Equal6
Equal4
Equal3
Equal0
movc~1_OUT0
movb~0_OUT0
mova~1_OUT0
jz~2_OUT0
jmp~6_OUT0
jmp~2_OUT0
jc~8_OUT0
nop~4
nop~5
nop~6
nop~7
nop~8
jz~7
jz~3
jz~4
sub~2
rsr~6
rsr~2
rsl~7
rsl~3
out1~8
out1~4
not1~4
not1~0
nop~9
jz~8
movc~2
movb~1
mova~2
jz~5
jmp~7
jmp~3
jc~9
sub
rsr
rsr~2_OUT0
rsl
rsl~3_OUT0
out1
out1~4_OUT0
not1
not1~0_OUT0
Equal6_OUT
nop
jz
movc
movb
mova
jz~5_OUT0
jmp
jmp~3_OUT0
jc



视图分析及结论：

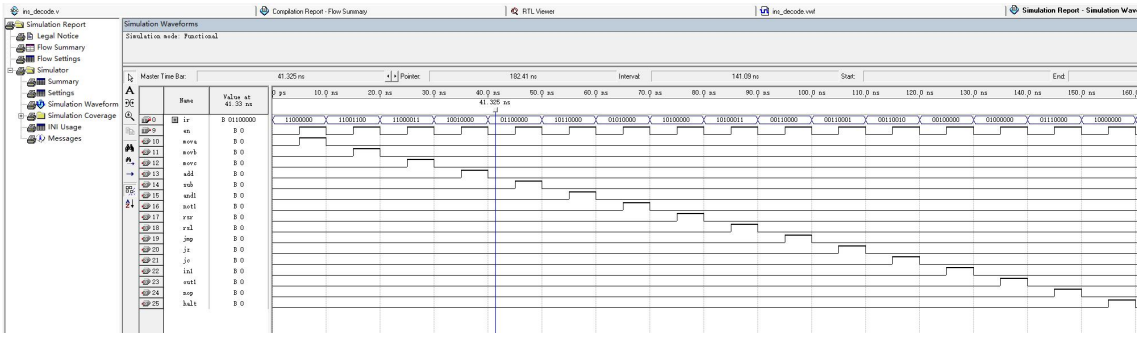
分析：视图左边为输入，右边为输出。其中连接有大量的元器件。例如比较器：当输入相等时输出 1，不相等时输出 0；还由大量的 2-1 选择器构成，当控制信号为 0 时，输出第一位，控制信号 1 时，输出第二位。图中输入信号为 ir 和 en，输出信号包括 add 等 16 种情况。各个输出端口之间通过导线相连。

结论：指令译码器功能需要经过多重门的处理后才能实现，其内部原理结构图十分复杂。

E) 功能仿真波形

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】，在【assignments】-【setting】-【simulator settings】-【simulation mode】中，将 timing 修改为 functional，然后再点击【processing】-【Generate Functional Simulation Netlist】，接着设置输入波形，成功后点击【start simulation】开始仿真，查看输出功能仿真波形图。

功能仿真波形图：



结果分析及结论：

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。

当 en 为 0 时，不管 ir 为何值，16 个输出全为 0

当 en 为 1 时：

当 ir=11000000 时，mova 输出为 1；

当 ir=11001100 时，movb 输出为 1；

当 ir=11000011 时，movc 输出为 1；

当 ir=10010000 时，add 输出为 1；

当 ir=01100000 时，sub 输出为 1；

当 ir=10110000 时，and1 输出为 1；

当 ir=01010000 时，not1 输出为 1；

当 ir=10100000 时，rsr 输出为 1；

当 ir=10100011 时，rsl 输出为 1；

当 ir=00110000 时，jmp 输出为 1；

当 ir=00110001 时，jz 输出为 1；

当 ir=00110010 时，jc 输出为 1；

当 ir=00100000 时，in1 输出为 1；

当 ir=01000000 时，out1 输出为 1；

当 ir=01110000 时，nop 输出为 1；

当 ir=10000000 时，halt 输出为 1；

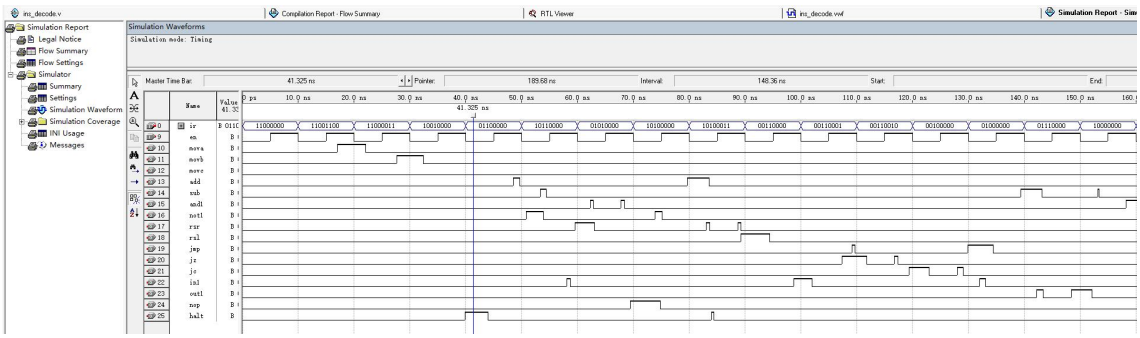
结论：功能仿真是指在一个设计中，在设计实现前对所创建的逻辑进行的验证其功能是否正确的过程。 布局布线以前的仿真都称作功能仿真，它包括综合前仿真（Pre-Synthesis

Simulation）和综合后仿真（Post-Synthesis Simulation）。 综合前仿真主要针对基于原理框图的设计 ；综合后仿真既适合原理图设计 ，也适合基于 HDL 语言的设计。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：

结果分析：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：时序仿真使用布局布线后器件给出的模块和连线的延时信息，在最坏的情况下对电路的行为作出实际地估价。时序仿真使用的仿真器和功能仿真使用的仿真器是相同的，所需的流程和激励也是相同的;唯一的差别是为时序仿真加载到仿真器的设计包括基于实际布局布线设计的最坏情况的布局布线延时，并且在仿真结果波形图中，时序仿真后的信号加载了时延，而功能仿真没有。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	15.130 ns	en	not1	--	--	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	15.130 ns	en	not1
2	N/A	None	15.089 ns	ir[3]	nop
3	N/A	None	15.033 ns	ir[2]	nop
4	N/A	None	14.811 ns	ir[5]	nop
5	N/A	None	14.800 ns	ir[4]	nop
6	N/A	None	14.768 ns	ir[3]	jmp
7	N/A	None	14.731 ns	ir[3]	halt
8	N/A	None	14.712 ns	ir[2]	jmp
9	N/A	None	14.678 ns	ir[2]	halt
10	N/A	None	14.490 ns	ir[5]	jmp
11	N/A	None	14.479 ns	ir[4]	jmp
12	N/A	None	14.470 ns	en	nop
13	N/A	None	14.455 ns	en	rsl
14	N/A	None	14.436 ns	ir[5]	rsl
15	N/A	None	14.426 ns	en	sub
16	N/A	None	14.400 ns	ir[5]	halt
17	N/A	None	14.344 ns	ir[3]	jc
18	N/A	None	14.332 ns	ir[5]	add
19	N/A	None	14.288 ns	ir[2]	jc
20	N/A	None	14.155 ns	ir[4]	not1
21	N/A	None	14.149 ns	en	jmp
22	N/A	None	14.141 ns	en	halt
23	N/A	None	14.066 ns	ir[5]	jc
24	N/A	None	14.055 ns	ir[4]	jc
25	N/A	None	14.032 ns	ir[4]	rsl
26	N/A	None	13.987 ns	ir[4]	add
27	N/A	None	13.960 ns	ir[4]	halt
28	N/A	None	13.906 ns	ir[5]	not1
29	N/A	None	13.727 ns	en	rsr
30	N/A	None	13.536 ns	en	add
31	N/A	None	13.406 ns	en	out1
32	N/A	None	13.359 ns	ir[4]	sub
33	N/A	None	13.239 ns	ir[3]	movc
34	N/A	None	13.195 ns	ir[5]	sub
35	N/A	None	13.193 ns	ir[2]	movc
36	N/A	None	13.050 ns	en	in1
37	N/A	None	13.047 ns	ir[5]	movc
38	N/A	None	12.998 ns	ir[4]	rsr
39	N/A	None	12.990 ns	en	jc
40	N/A	None	12.834 ns	ir[5]	rsr
41	N/A	None	12.801 ns	en	and1
42	N/A	None	12.782 ns	ir[5]	and1
43	N/A	None	12.686 ns	en	movc
44	N/A	None	12.683 ns	ir[2]	movb
45	N/A	None	12.656 ns	ir[4]	movc
46	N/A	None	12.653 ns	en	movb
47	N/A	None	12.625 ns	ir[4]	movb
48	N/A	None	12.621 ns	ir[3]	jz
49	N/A	None	12.565 ns	ir[2]	jz
50	N/A	None	12.549 ns	ir[3]	mova
51	N/A	None	12.545 ns	ir[5]	movb
52	N/A	None	12.503 ns	ir[2]	mova
53	N/A	None	12.355 ns	ir[3]	movb
54	N/A	None	12.343 ns	ir[5]	jz
55	N/A	None	12.341 ns	ir[4]	out1
56	N/A	None	12.339 ns	ir[4]	and1
57	N/A	None	12.332 ns	ir[4]	jz
58	N/A	None	12.273 ns	ir[5]	mova
59	N/A	None	12.185 ns	ir[5]	out1
60	N/A	None	12.017 ns	en	mova
61	N/A	None	11.993 ns	ir[4]	in1
62	N/A	None	11.882 ns	ir[4]	mova
63	N/A	None	11.781 ns	ir[5]	in1
64	N/A	None	11.777 ns	en	jz
65	N/A	None	10.787 ns	ir[7]	not1
66	N/A	None	10.190 ns	ir[1]	nop
67	N/A	None	10.083 ns	ir[7]	sub
68	N/A	None	10.064 ns	ir[6]	rsl
69	N/A	None	10.005 ns	ir[0]	nop
70	N/A	None	9.900 ns	ir[7]	rsl
71	N/A	None	9.869 ns	ir[1]	jmp
72	N/A	None	9.861 ns	ir[1]	halt
73	N/A	None	9.847 ns	ir[7]	halt
74	N/A	None	9.740 ns	ir[6]	halt
75	N/A	None	9.708 ns	ir[6]	nop
76	N/A	None	9.704 ns	ir[7]	add
77	N/A	None	9.699 ns	ir[7]	rsr
78	N/A	None	9.684 ns	ir[0]	jmp
79	N/A	None	9.676 ns	ir[0]	halt
80	N/A	None	9.674 ns	ir[6]	not1
81	N/A	None	9.639 ns	ir[6]	add
82	N/A	None	9.592 ns	ir[6]	rsr
83	N/A	None	9.464 ns	ir[6]	jc
84	N/A	None	9.447 ns	ir[1]	rsr
85	N/A	None	9.432 ns	ir[7]	nop
86	N/A	None	9.408 ns	ir[0]	jc
87	N/A	None	9.308 ns	ir[6]	jmp
88	N/A	None	9.262 ns	ir[0]	rsl
89	N/A	None	9.262 ns	ir[0]	rsr
90	N/A	None	9.236 ns	ir[7]	jc
91	N/A	None	9.112 ns	ir[7]	jmp
92	N/A	None	9.090 ns	ir[1]	rsl
93	N/A	None	9.063 ns	ir[7]	out1
94	N/A	None	9.053 ns	ir[1]	jc
95	N/A	None	8.945 ns	ir[6]	sub
96	N/A	None	8.707 ns	ir[7]	in1
97	N/A	None	8.513 ns	ir[6]	movc
98	N/A	None	8.498 ns	ir[6]	jz
99	N/A	None	8.410 ns	ir[6]	and1
100	N/A	None	8.311 ns	ir[1]	jz
101	N/A	None	8.271 ns	ir[1]	movc
102	N/A	None	8.248 ns	ir[6]	movb
103	N/A	None	8.246 ns	ir[7]	and1
104	N/A	None	8.153 ns	ir[7]	jz
105	N/A	None	8.122 ns	ir[7]	movc
106	N/A	None	8.080 ns	ir[7]	movb
107	N/A	None	7.935 ns	ir[6]	out1
108	N/A	None	7.819 ns	ir[1]	mova
109	N/A	None	7.663 ns	ir[0]	movc
110	N/A	None	7.548 ns	ir[0]	mova
111	N/A	None	7.541 ns	ir[6]	in1
112	N/A	None	7.417 ns	ir[7]	mova
113	N/A	None	7.347 ns	ir[0]	jz
114	N/A	None	6.721 ns	ir[6]	mova

Tpd 图

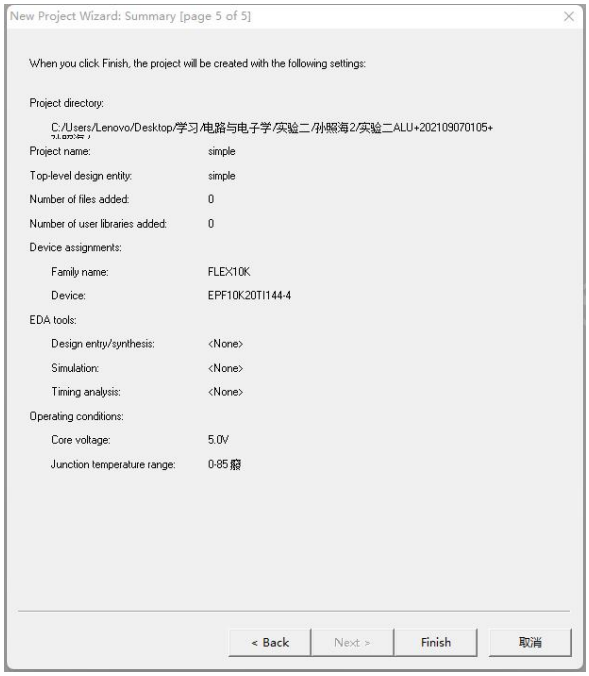
结果分析及结论：

结果分析：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 en 到 not1 的最坏定时情况的 tpd 为 15.130ns。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 ir[3]到 nop 的 tpd 为 15.089ns。

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、算术逻辑单元 ALU

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



B）编写源代码

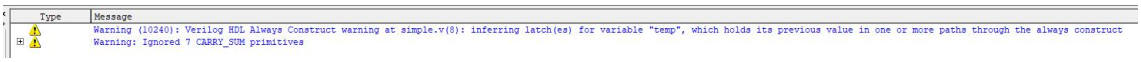
Verilog1.v*

```
1  module simple(m,s,a,b,t,cf,zf);
2    input wire m;
3    input wire [3:0] s;
4    input wire [7:0] a,b;
5    output reg [7:0] t;
6    output reg cf,zf;
7    reg[8:0] temp; //非输入输出
8    always @(m or s or a or b)
9    begin //当m或s或a或b有一个不为1时
10     {cf,zf}=0;
11     t=00000000;
12     if(m==1'b0)
13     begin
14     if(s[3:0]==4'b1100||s[3:0]==4'b0100) t=a;
15       else if(s[3:0]==4'b1010) t=b;
16     end
17     else if(m==1'b1)
18     begin
19       if(s[3:0]==4'b1001)
20       begin
21         t=a+b;
22         if(t==8'b00000000) zf=1;
23         else zf=0;
24         temp=a+b;
25         if(temp[8]==1'b0) cf=0;
26         else cf=1;
27       end
28         else if(s[3:0]==4'b0110)
29         begin
30           t=b-a;
31           if(t==8'b00000000) zf=1;
32           else zf=0;
33           temp=b-a;
34           if(temp[8]==1'b0) cf=0;
35           else cf=1;
36         end
37         else if(s[3:0]==4'b1011) t=a&b;
38         else if(s[3:0]==4'b0101) t=~b;
39     end
40   end
41
42 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）

无错误。

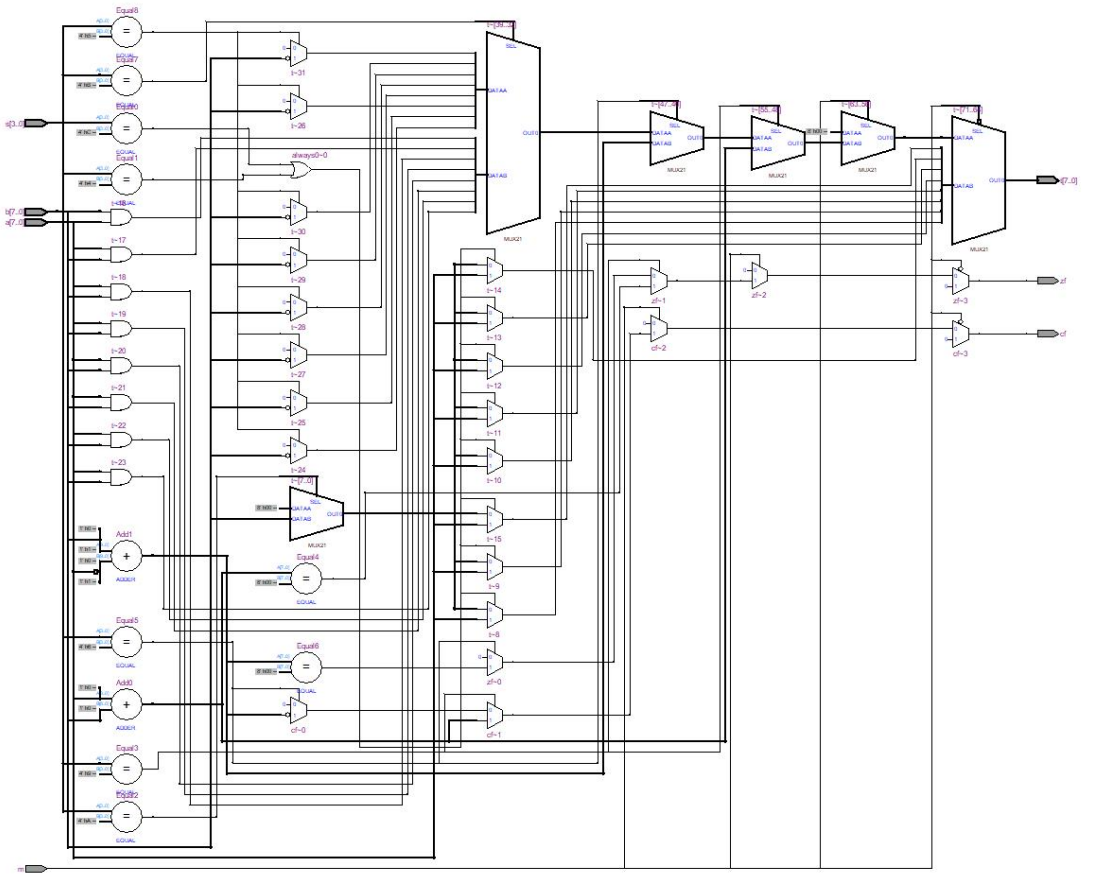
警告信息：



资源消耗：

Flow Status	Successful - Wed Nov 16 23:43:24 2022
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	simple
Top-level Entity Name	simple
Family	FLEX10K
Device	EPF10K20TI144-4
Timing Models	Final
Met timing requirements	Yes
Total logic elements	72 / 1,152 (6 %)
Total pins	31 / 102 (30 %)
Total memory bits	0 / 12,288 (0 %)

D) RTL 视图



结果分析及结论：

结果分析：视图左边为输入，右边为输出。其中连接有大量的元器件。比如比较器：当输入相等时输出 1，不相等时输出 0；还由大量的 2-1 选择器构成，当控制信号为 0 时，输出第

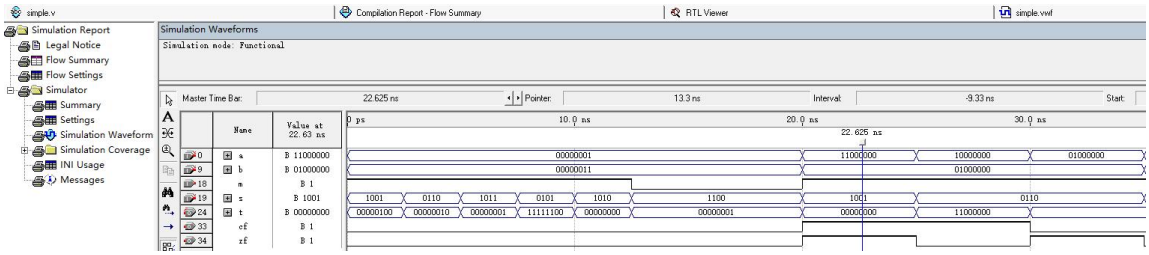
一位，控制信号为 1 时，输出第二位。图中输入信号为 m，s，a，b，输出信号为 t，cf，zf。各个输出端口之间通过导线相连。

结论：ALU 需要经过多重门的处理后才能实现，ALU 的内部原理结构图十分复杂。

E）功能仿真波形

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】，在【assignments】-【setting】-【simulator settings】-【simulation mode】中，将 timing 修改为 functional，然后再点击【processing】-【Generate Functional Simulation Netlist】，接着设置输入波形，成功后点击【start simulation】开始仿真，查看输出功能仿真波形图。

功能仿真波形图：



结果分析及结论：

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。

当控制信号 m 为 1，s 为 1001 时，执行 t=a+b

当控制信号 m 为 1，s 为 0110 时，执行 t=b-a

当控制信号 m 为 1，s 为 1011 时，执行 t=a&b

当控制信号 m 为 1，s 为 0101 时，执行 t=~b

当控制信号 m 为 0，s 为 1010 时，执行 t=b

当控制信号 m 为 0，s 为 1100 或 0100 时，执行 t=a

有进位和借位时 cf 为 1，否则为 0；

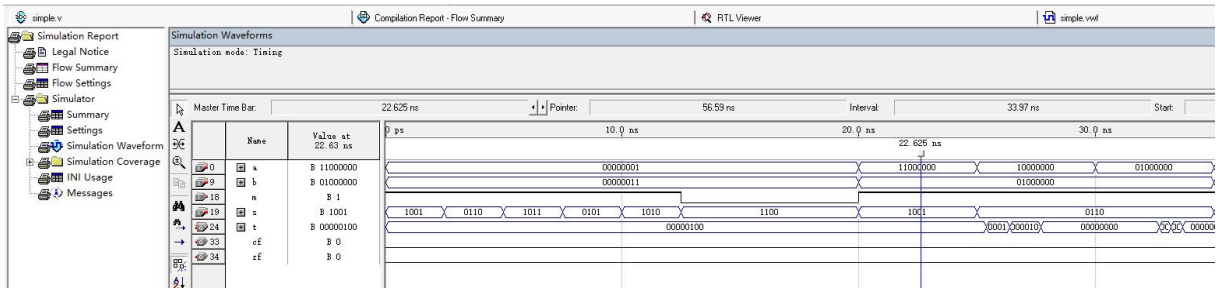
和为 0 或差为 0 时 zf 为 1，否则为 0；

结论：功能仿真是指在一个设计中，在设计实现前对所创建的逻辑进行的验证其功能是否正确的过程。 布局布线以前的仿真都称作功能仿真，它包括综合前仿真（Pre-Synthesis Simulation）和综合后仿真（Post-Synthesis Simulation）。 综合前仿真主要针对基于原理框图的设计 ；综合后仿真既适合原理图设计 ，也适合基于 HDL 语言的设计。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：

结果分析：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：时序仿真使用布局布线后器件给出的模块和连线的延时信息，在最坏的情况下对电路的行为作出实际地估价。时序仿真使用的仿真器和功能仿真使用的仿真器是相同的，所需的流程和激励也是相同的;唯一的差别是为时序仿真加载到仿真器的设计包括基于实际布局布线设计的最坏情况的布局布线延时，并且在仿真结果波形图中，时序仿真后的信号加载了时延，而功能仿真没有。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	52.900 ns	b[1]	zf	--	--	0
2	Total number of failed paths								0

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	52.900 ns	b[1]	zf
2	N/A	None	52.100 ns	a[0]	zf
3	N/A	None	51.600 ns	b[0]	zf
4	N/A	None	51.600 ns	a[1]	zf
5	N/A	None	48.700 ns	b[2]	zf
6	N/A	None	48.600 ns	a[2]	zf
7	N/A	None	46.100 ns	b[1]	t[7]
8	N/A	None	45.800 ns	b[1]	cf
9	N/A	None	45.300 ns	a[0]	t[7]
10	N/A	None	45.000 ns	a[0]	cf
11	N/A	None	44.800 ns	b[0]	t[7]
12	N/A	None	44.800 ns	a[1]	t[7]
13	N/A	None	44.500 ns	b[0]	cf
14	N/A	None	44.500 ns	a[1]	cf
15	N/A	None	44.200 ns	b[3]	zf
16	N/A	None	42.700 ns	a[3]	zf
17	N/A	None	41.900 ns	b[2]	t[7]
18	N/A	None	41.800 ns	a[2]	t[7]
19	N/A	None	41.600 ns	b[2]	cf
20	N/A	None	41.500 ns	a[2]	cf
21	N/A	None	40.100 ns	b[1]	t[6]
22	N/A	None	39.400 ns	a[4]	zf
23	N/A	None	39.300 ns	a[0]	t[6]
24	N/A	None	38.900 ns	b[4]	zf
25	N/A	None	38.800 ns	b[0]	t[6]
26	N/A	None	38.800 ns	a[1]	t[6]
27	N/A	None	38.300 ns	b[1]	t[5]
28	N/A	None	37.500 ns	a[0]	t[5]
29	N/A	None	37.400 ns	b[3]	t[7]
30	N/A	None	37.100 ns	b[3]	cf
31	N/A	None	37.000 ns	b[0]	t[5]
32	N/A	None	37.000 ns	a[1]	t[5]
33	N/A	None	35.900 ns	a[3]	t[7]
34	N/A	None	35.900 ns	b[2]	t[6]
35	N/A	None	35.800 ns	a[2]	t[6]
36	N/A	None	35.600 ns	a[3]	cf
37	N/A	None	34.700 ns	b[5]	zf
38	N/A	None	34.500 ns	a[5]	zf
39	N/A	None	34.100 ns	b[2]	t[5]
40	N/A	None	34.000 ns	a[2]	t[5]
41	N/A	None	32.600 ns	a[4]	t[7]
42	N/A	None	32.300 ns	a[4]	cf
43	N/A	None	32.100 ns	a[6]	zf
44	N/A	None	32.100 ns	b[4]	t[7]
45	N/A	None	32.100 ns	b[1]	t[4]
46	N/A	None	31.800 ns	b[6]	zf
47	N/A	None	31.800 ns	b[4]	cf
48	N/A	None	31.400 ns	b[3]	t[6]
49	N/A	None	31.300 ns	a[0]	t[4]
50	N/A	None	30.800 ns	b[0]	t[4]
51	N/A	None	30.800 ns	a[1]	t[4]
52	N/A	None	29.900 ns	a[3]	t[6]
53	N/A	None	29.600 ns	b[3]	t[5]
54	N/A	None	28.100 ns	s[3]	zf
55	N/A	None	28.100 ns	a[3]	t[5]
56	N/A	None	27.900 ns	b[5]	t[7]
57	N/A	None	27.900 ns	b[2]	t[4]
58	N/A	None	27.800 ns	a[7]	zf
59	N/A	None	27.800 ns	a[2]	t[4]
60	N/A	None	27.700 ns	s[0]	zf
61	N/A	None	27.700 ns	a[5]	t[7]
62	N/A	None	27.700 ns	b[1]	t[3]
63	N/A	None	27.600 ns	b[5]	cf
64	N/A	None	27.400 ns	a[5]	cf
65	N/A	None	26.900 ns	a[0]	t[3]
66	N/A	None	26.800 ns	s[3]	t[5]
67	N/A	None	26.800 ns	s[3]	t[2]
68	N/A	None	26.600 ns	s[2]	zf
69	N/A	None	26.600 ns	s[1]	zf
70	N/A	None	26.600 ns	b[7]	zf
71	N/A	None	26.600 ns	a[4]	t[6]
72	N/A	None	26.400 ns	b[0]	t[3]
73	N/A	None	26.400 ns	a[1]	t[3]
74	N/A	None	26.300 ns	s[3]	t[0]
75	N/A	None	26.200 ns	s[3]	t[1]
76	N/A	None	26.100 ns	s[3]	t[7]
77	N/A	None	26.100 ns	s[3]	t[6]
78	N/A	None	26.100 ns	b[4]	t[6]
79	N/A	None	26.100 ns	s[3]	t[4]
80	N/A	None	25.900 ns	s[0]	t[5]
81	N/A	None	25.900 ns	s[0]	t[2]
82	N/A	None	25.400 ns	s[0]	t[0]
83	N/A	None	25.300 ns	a[6]	t[7]
84	N/A	None	25.300 ns	s[0]	t[1]
85	N/A	None	25.200 ns	s[0]	t[7]
86	N/A	None	25.200 ns	s[0]	t[6]
87	N/A	None	25.200 ns	s[0]	t[4]
88	N/A	None	25.000 ns	a[6]	cf
89	N/A	None	25.000 ns	b[6]	t[7]
90	N/A	None	24.800 ns	m	zf
91	N/A	None	24.800 ns	s[3]	cf
92	N/A	None	24.800 ns	s[2]	t[5]
93	N/A	None	24.800 ns	a[4]	t[5]
94	N/A	None	24.800 ns	s[2]	t[2]
95	N/A	None	24.700 ns	b[6]	cf
96	N/A	None	24.300 ns	s[1]	t[5]
97	N/A	None	24.300 ns	b[4]	t[5]
98	N/A	None	24.300 ns	s[1]	t[2]
99	N/A	None	24.300 ns	s[2]	t[0]
100	N/A	None	24.200 ns	s[2]	t[1]
101	N/A	None	24.100 ns	s[2]	t[7]
102	N/A	None	24.100 ns	s[2]	t[6]
103	N/A	None	24.100 ns	s[2]	t[4]
104	N/A	None	23.900 ns	s[0]	cf
105	N/A	None	23.800 ns	s[3]	t[3]
106	N/A	None	23.800 ns	b[1]	t[2]
107	N/A	None	23.800 ns	s[1]	t[0]
108	N/A	None	23.700 ns	s[1]	t[1]
109	N/A	None	23.600 ns	s[1]	t[7]
110	N/A	None	23.600 ns	s[1]	t[6]
111	N/A	None	23.600 ns	s[1]	t[4]
112	N/A	None	23.500 ns	b[3]	t[4]
113	N/A	None	23.500 ns	b[2]	t[3]
114	N/A	None	23.400 ns	a[2]	t[3]
115	N/A	None	23.000 ns	a[7]	t[7]
116	N/A	None	23.000 ns	a[0]	t[2]
117	N/A	None	23.000 ns	b[2]	t[2]
118	N/A	None	22.900 ns	a[6]	t[6]
119	N/A	None	22.900 ns	s[0]	t[3]
120	N/A	None	22.800 ns	s[2]	cf
121	N/A	None	22.800 ns	a[2]	t[2]
122	N/A	None	22.700 ns	b[1]	t[1]
123	N/A	None	22.500 ns	b[6]	t[6]
124	N/A	None	22.500 ns	b[3]	t[3]
125	N/A	None	22.500 ns	b[0]	t[2]
126	N/A	None	22.500 ns	a[1]	t[2]
127	N/A	None	22.300 ns	s[1]	cf
128	N/A	None	22.300 ns	b[5]	t[5]
129	N/A	None	22.100 ns	a[5]	t[5]
130	N/A	None	22.000 ns	a[3]	t[4]
131	N/A	None	21.900 ns	b[5]	t[6]
132	N/A	None	21.900 ns	a[4]	t[4]
133	N/A	None	21.800 ns	s[2]	t[3]
134	N/A	None	21.700 ns	b[7]	t[7]
135	N/A	None	21.700 ns	a[5]	t[6]
136	N/A	None	21.700 ns	s[1]	t[3]
137	N/A	None	21.600 ns	m	t[7]
138	N/A	None	21.600 ns	m	t[4]
139	N/A	None	21.600 ns	m	t[2]
140	N/A	None	21.500 ns	m	t[3]
141	N/A	None	21.400 ns	m	t[6]
142	N/A	None	21.400 ns	m	t[5]
143	N/A	None	21.400 ns	b[4]	t[4]
144	N/A	None	21.300 ns	m	t[1]
145	N/A	None	21.300 ns	a[1]	t[1]
146	N/A	None	21.000 ns	a[3]	t[3]
147	N/A	None	20.900 ns	m	t[0]
148	N/A	None	20.700 ns	a[7]	cf
149	N/A	None	20.200 ns	m	cf
150	N/A	None	19.900 ns	a[0]	t[0]
151	N/A	None	19.900 ns	b[0]	t[0]
152	N/A	None	19.700 ns	a[0]	t[1]
153	N/A	None	19.700 ns	b[0]	t[1]
154	N/A	None	19.200 ns	b[7]	cf

结果分析及结论：

结果分析：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[1]到 zf 的最坏定时情况的 tpd 为 52.900ns。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[0]到 zf 的 tpd 为

52.100ns。

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．指令译码器必须要 16 个输出吗？可否将一些输出合并，哪些可以合并，为什么？
不需要。jmp 和 add 可以合并起来，因为 jmp 是将 add 后的结果写入 pc 中，则可以进行 add 操作后直接进行写入操作。add 和 sub 和 and 操作可以合并，因为这三个操作类似，且输出为使能信号，故可以用一个合并使能信号来作为三个输出的共同使能信号。

2．ALU 中的 S[3..0]控制信号是来自哪里或者说与什么信息相同？
来自指令码 ir 的前四位，即 ir[7:4]；

3、为何 S[3..0]等于 1100 或 0100 时将输入 a 传给 t，S[3..0]等于 1010 时将输入 b 传给 t？
S[3..0]为控制信号，当 S[3..0]输入为 1100 或 0100 时，控制输出 t 等于 a，S[3..0]等于 1010 时，t 等于 b，此时 ALU 相当于选择器。

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。

基本了解了简易模型机的内部结构。同时熟悉了译码器、ALU 的工作原理。学会使用 Verilog 语言描述简单的电路。但也遇到了一些困难，比如开始不懂如何使用 QuartusII 进行仿真。但通过上网查询相关资料和询问同学后得以解决问题，并通过分析仿真实验发现电路中的问题。还有不懂指令译码器和 ALU 的工作原理等，但通过上网查询资料和小班讨论课讨论后基本都得以解决。以后再遇到相关问题时，可以通过先上网查询资料，再询问老师和同学来解决。

2、对本实验内容、过程和方法的改进建议（可选项）。

可以给同学们演示一下如何使用 QuartusII 进行 Verilog 文件的编译功能仿真和时序仿真。例如出一个演示视频，网上虽然能找到比较类似的资源，但还不是特别相似，需要耗费大量时间寻找、学习这些基础操作。