

实验二 模型机组合部件的实现（一）

班级 智能 2103 班 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. 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

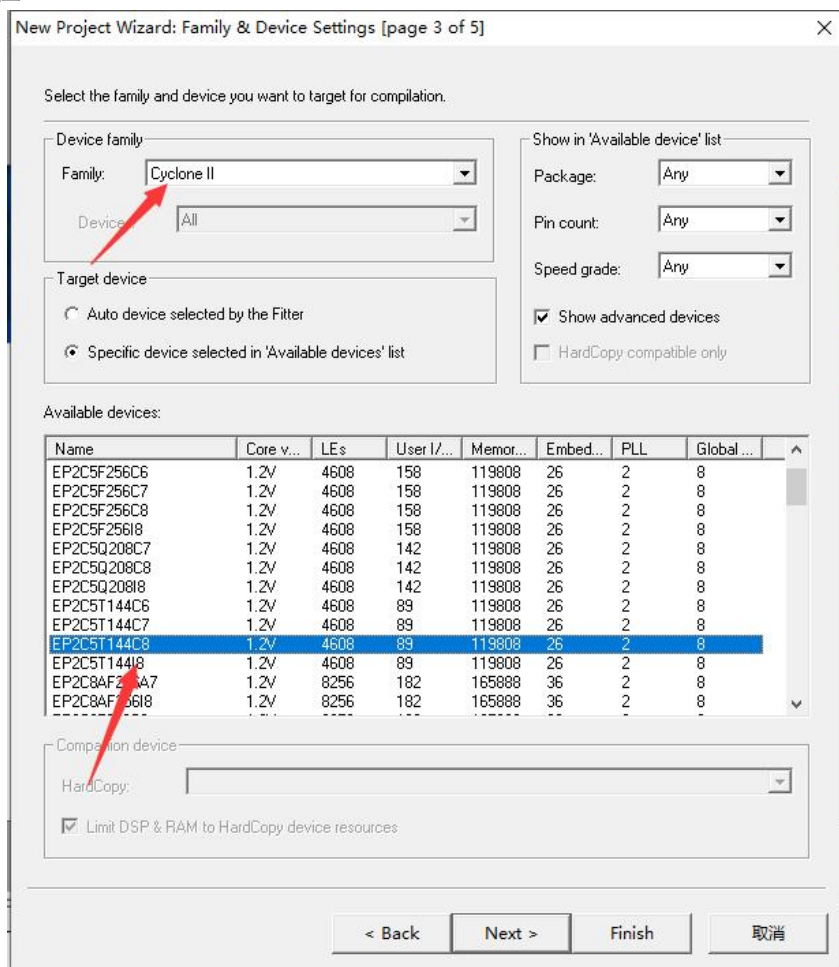
1. 用 VERILOG 语言设计指令译码器；
2. 用 VERILOG 语言设计 ALU。

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）

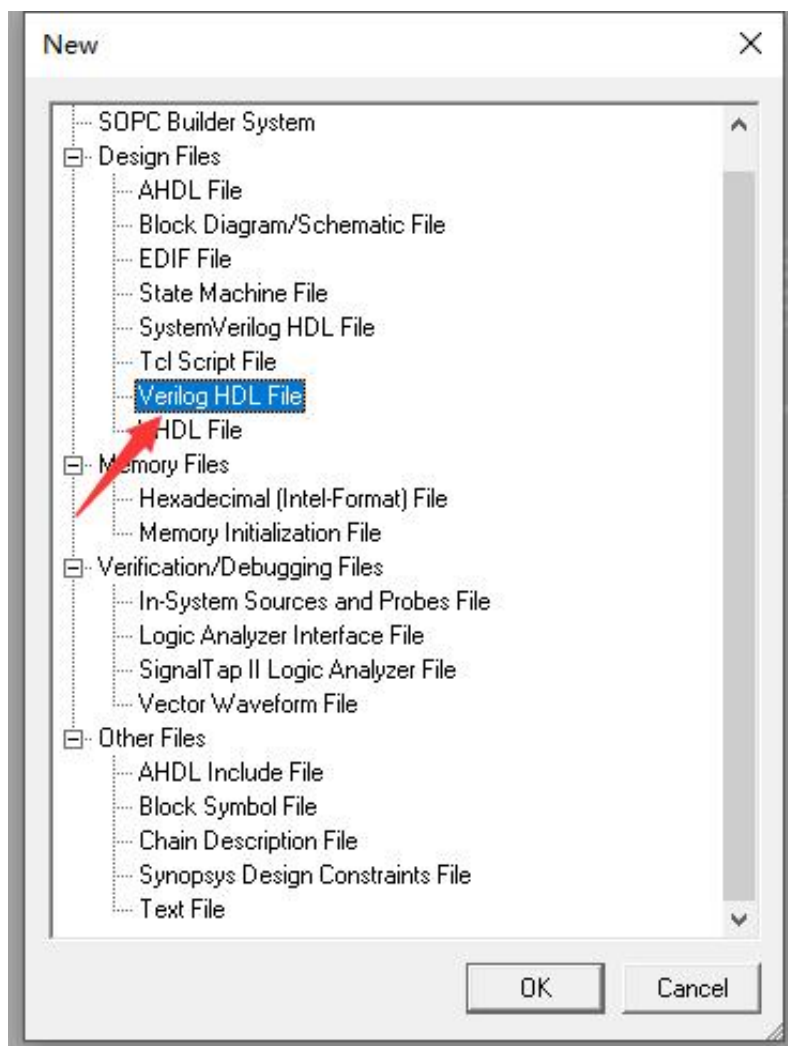
步骤：左上角 file->New Project Wizard->选择工程位置和工程名->选择芯片 Cyclone II, available device 中选择 EP2C5T144C8->点击 next->最后点击 finish 完成创建工程



B) 编写源代码

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



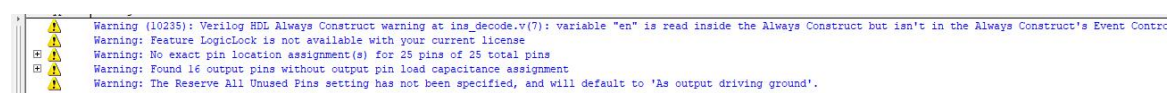
```

1 module decoder(
2     input wire en,
3     input wire [7:0] ir,
4     output reg mova,movb,movc,add,sub,andl,notl,rsr,rs1,jmp,jz,jc,inl,outl,nop,halt);
5 always @(ir) begin //sensitive list begin--end
6     [mova,movb,movc,add,sub,andl,notl,rsr,rs1,jmp,jz,jc,inl,outl,nop,halt]=0;//initialize
7     if(en) begin
8         if(ir[7:4]==4'b1100) begin//12
9             if(ir[3:2]==2'b11) begin mova=0;movb=1;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
10            else if(ir[1:0]==2'b11) begin mova=0;movb=0;movc=1;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
11            else begin mova=1;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
12        end
13        else if(ir[7:4]==4'b1001) begin mova=0;movb=0;movc=0;add=1;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
14        else if(ir[7:4]==4'b0110) begin mova=0;movb=0;movc=0;add=0;sub=1;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
15        else if(ir[7:4]==4'b0101) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=1;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
16        else if(ir[7:4]==4'b0100) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=1;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
17        else if(ir[7:4]==4'b1010) begin//10
18            if(ir[1:0]==2'b00) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=1;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
19            else if(ir[1:0]==2'b11) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=1;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end//3
20        end
21        else if(ir[7:2]==6'b001100) begin //12
22            if(ir[1:0]==2'b00) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=1;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end//0
23            else if(ir[1:0]==2'b01) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=1;jc=0;inl=0;outl=0;nop=0;halt=0;end//1
24            else if(ir[1:0]==2'b10) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=1;inl=0;outl=0;nop=0;halt=0;end//2
25        end
26        else if(ir[7:4]==4'b0010) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=1;outl=0;nop=0;halt=0;end//2
27        else if(ir[7:4]==4'b0100) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=1;nop=0;halt=0;end//4
28        else if(ir[7:4]==4'b0111) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=1;halt=0;end//7
29        else if(ir[7:4]==4'b1000) begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=1;end//8
30        else begin mova=0;movb=0;movc=0;add=0;sub=0;andl=0;notl=0;rsr=0;rs1=0;jmp=0;jz=0;jc=0;inl=0;outl=0;nop=0;halt=0;end
31    end
32 end
33 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 - Wed Nov 16 23:35:19 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 | 26 / 4,608 (< 1 %) |
| Total combinational functions | 26 / 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 %) |

视图分析及结论：

代码出现提示：Warning: No exact pin location assignment(s) for 5 pins of 5 total pins, 是因为没有分配管脚，由于此次实验没在实体电路板上运行，因此引脚可以暂时不做分配。

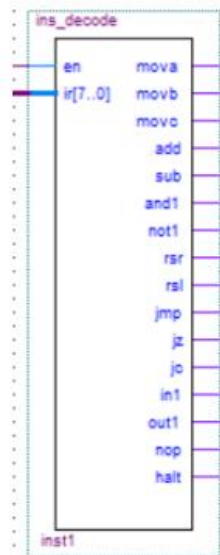
代码出现提示：Warning: Found 16 output pins without output pin load capacitance assignment, 是因为没有给输出管脚指定负载电容，该功能用于估算 TCO 和功耗，可以不理睬。

图中出现大量的选择器，是因为存在很多的条件判断语句，需要对信号进行分类判断从而决定输出。

图中没有出现锁存器，是因为初始化赋值都是 0，在每一种的输入下，所有的输出端口都有赋值。

一个功能的实现需要经过多重门的处理后才能实现，一个元件的内部原理结构图十分复杂。

接口设计：

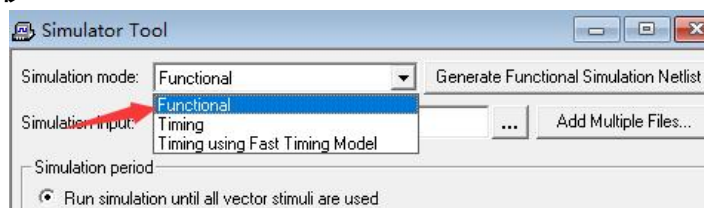


指令译码器的输入输出引脚如上图所示。en 为使能信号，ir[7..0]是 8 位指令编码，输出是对应的 16 条指令。

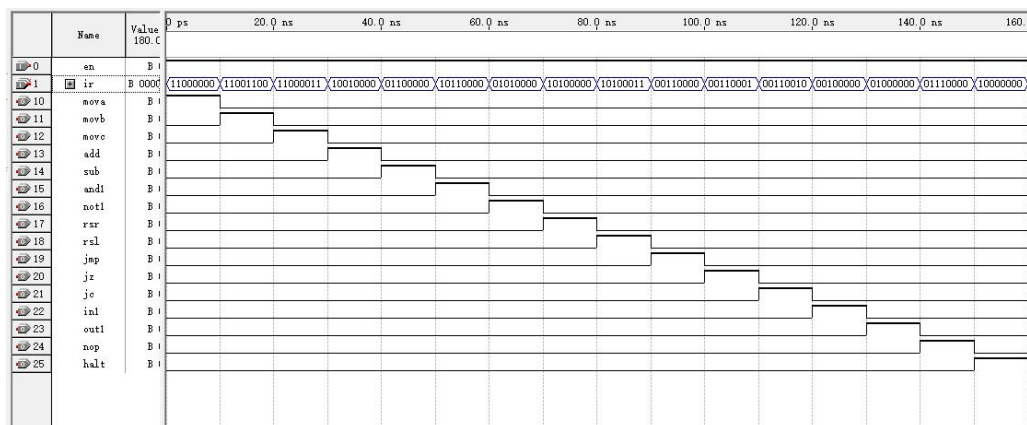
指令译码器是根据指令系统表中的指令编码，对输入的 8 位指令进行解析，判定是哪条指令，则对应指令的输出为 1，否则输出为 0。

| 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 | (开关 7-0) $\rightarrow R1$ | 0010 R1 XX |
| OUT R2 | $(R2) \rightarrow$ 发光二极管 7-0 | 0100 XX R2 |
| NOP | | 0111 00 00 |
| HALT | 停机 | 1000 00 00 |

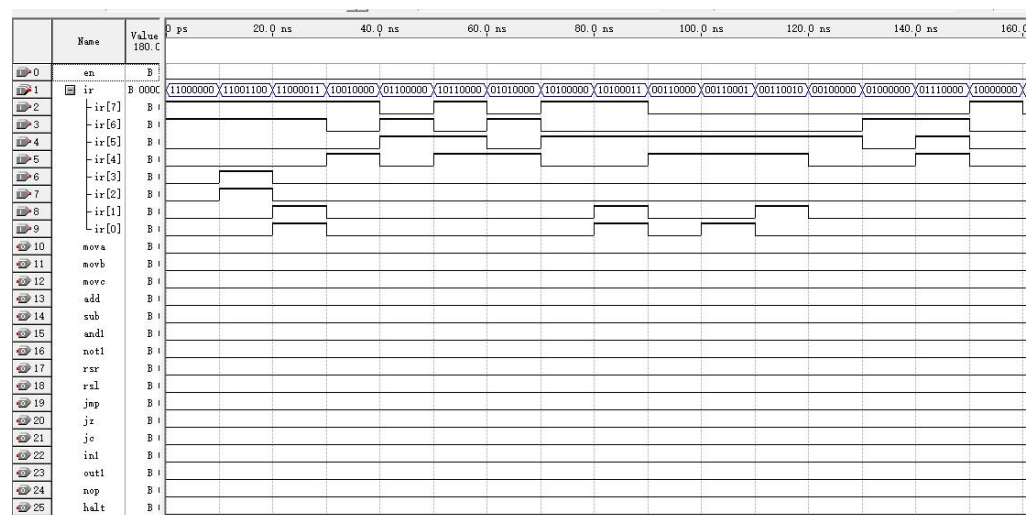
E) 功能仿真波形



使能为 1:



使能为 0:



结果分析及结论:

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。

功能仿真的缺点是不带有任何的门延时、线延时等等，只是理想情况下的仿真，优点是仿真速度快，可以根据需要观察电路输入输出端口和电路内部任一信号寄存器的波形。

使能信号为 1 时，指令可进入电路，按其译码逻辑，准确输出。

使能信号为 0 时，拒绝指令的进入，输出全部为 0。

当 en 为 0 时，不管 ir 为何值，16 个输出全为 0

当 en 为 1 时:

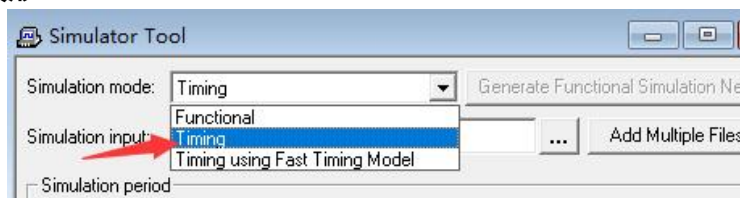
当 ir=11000000 时，movb 输出为 1;

当 ir=11001100 时，movb 输出为 1;

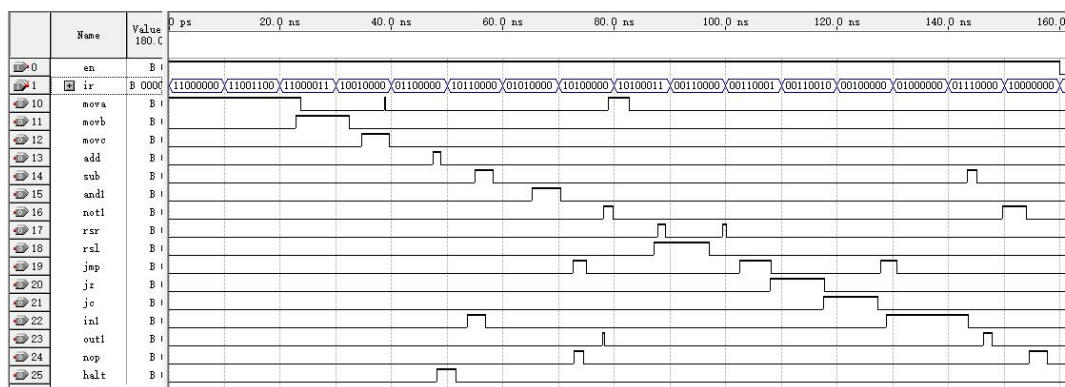
当 ir=11000011 时，movc 输出为 1;

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;

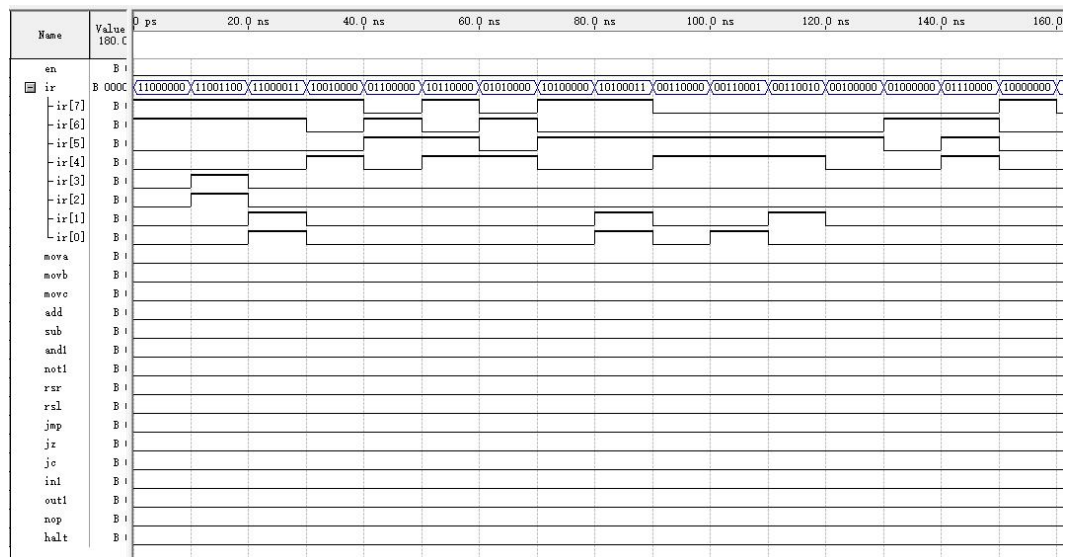
F) 时序仿真波形



使能 1:



使能 0:



结果分析及结论：

时序仿真存在延迟，因为时序仿真是在布线后进行，布线的延时信息，而且它和特定的器件是有关系的，又包含了器件的延时信息，所以是最接近真实器件运行的仿真。

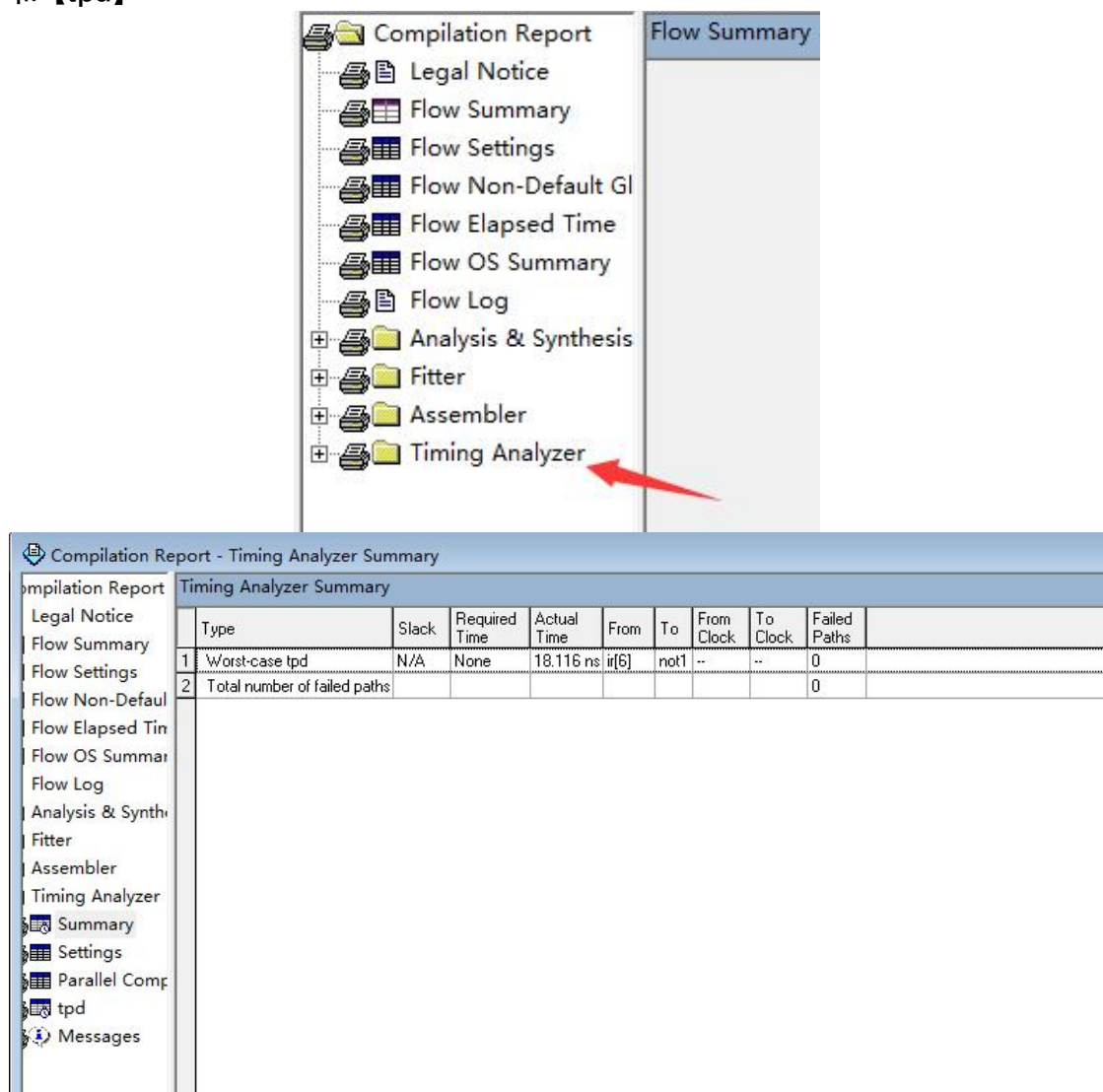
从波形中可以看出，当输入状态发生改变时，输出结果并未同时发生改变，而是有一定延迟，同时由于输入状态的改变，电路出现“冒险”，导致输出结果未与预期结果相同。但是由于只有使能为 1 时，输入信号才会存在不同路径对输出信号产生影响，因此只有使能为 1 时，才会出现冒险。当使能为 0 时，所有输出信号所有时间均为 0。

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；惟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。

G) 时序分析

操作方法是：编译后，在 compilation report 中选择【timing analysis】-【summary】和【tpd】



The screenshot shows the 'Compilation Report - Timing Analyzer Summary' window. The left pane lists various report items, with 'Timing Analyzer' expanded to show 'Summary', 'Settings', 'Parallel Comp', 'tpd', and 'Messages'. The right pane displays the 'Timing Analyzer Summary' table.

| Type | Slack | Required Time | Actual Time | From | To | From Clock | To Clock | Failed Paths |
|--------------------------------|-------|---------------|-------------|-------|------|------------|----------|--------------|
| 1 Worst-case tpd | N/A | None | 18.116 ns | ir[6] | not1 | -- | -- | 0 |
| 2 Total number of failed paths | | | | | | | | 0 |

实验一 译码器的实现

| Compilation Report | tpd | | | | | | |
|----------------------|-----|-------|-------------------|-----------------|-------|------|--|
| Legal Notice | | Slack | Required P2P Time | Actual P2P Time | From | To | |
| Flow Summary | 1 | N/A | None | 18.116 ns | ir[6] | not1 | |
| Flow Settings | 2 | N/A | None | 17.879 ns | ir[6] | rsr | |
| Flow Non-Default | 3 | N/A | None | 17.864 ns | en | not1 | |
| Flow Elapsed Time | 4 | N/A | None | 17.808 ns | ir[6] | jmp | |
| Flow OS Summary | 5 | N/A | None | 17.700 ns | ir[6] | jz | |
| Flow Log | 6 | N/A | None | 17.581 ns | en | rsr | |
| Analysis & Synthesis | 7 | N/A | None | 17.559 ns | en | jmp | |
| Fitter | 8 | N/A | None | 17.487 ns | ir[6] | add | |
| Assembler | 9 | N/A | None | 17.451 ns | en | jz | |
| Timing Analyzer | 10 | N/A | None | 17.396 ns | ir[6] | movc | |
| Summary | 11 | N/A | None | 17.215 ns | ir[6] | jc | |
| Settings | 12 | N/A | None | 17.189 ns | en | add | |
| Parallel Comp | 13 | N/A | None | 17.144 ns | en | movc | |
| tpd | 14 | N/A | None | 16.966 ns | en | jc | |
| Messages | 15 | N/A | None | 16.932 ns | ir[5] | and1 | |
| | 16 | N/A | None | 16.790 ns | ir[6] | in1 | |
| | 17 | N/A | None | 16.729 ns | ir[6] | halt | |
| | 18 | N/A | None | 16.541 ns | en | in1 | |
| | 19 | N/A | None | 16.431 ns | en | halt | |
| | 20 | N/A | None | 16.258 ns | ir[6] | out1 | |
| | 21 | N/A | None | 16.214 ns | ir[6] | movc | |
| | 22 | N/A | None | 16.006 ns | en | out1 | |
| | 23 | N/A | None | 15.962 ns | en | movc | |
| | 24 | N/A | None | 15.718 ns | ir[6] | movb | |
| | 25 | N/A | None | 15.466 ns | en | movb | |
| | 26 | N/A | None | 15.154 ns | ir[6] | and1 | |
| | 27 | N/A | None | 15.107 ns | ir[3] | movc | |
| | 28 | N/A | None | 15.055 ns | ir[5] | jmp | |
| | 29 | N/A | None | 15.050 ns | ir[5] | sub | |
| | 30 | N/A | None | 14.947 ns | ir[5] | jz | |
| | 31 | N/A | None | 14.860 ns | ir[3] | jmp | |
| | 32 | N/A | None | 14.859 ns | en | and1 | |
| | 33 | N/A | None | 14.752 ns | ir[3] | jz | |
| | 34 | N/A | None | 14.684 ns | ir[2] | movc | |
| | 35 | N/A | None | 14.646 ns | ir[5] | movc | |
| | 36 | N/A | None | 14.527 ns | ir[5] | rsr | |
| | 37 | N/A | None | 14.475 ns | ir[5] | nop | |
| | 38 | N/A | None | 14.462 ns | ir[5] | jc | |
| | 39 | N/A | None | 14.419 ns | ir[5] | rsr | |

| Compilation Report | tpd | | | | | | |
|----------------------|-----|-------|-------------------|-----------------|-------|------|--|
| Legal Notice | | Slack | Required P2P Time | Actual P2P Time | From | To | |
| Flow Summary | 40 | N/A | None | 14.267 ns | ir[3] | jc | |
| Flow Settings | 41 | N/A | None | 14.178 ns | ir[2] | jmp | |
| Flow Non-Default | 42 | N/A | None | 14.070 ns | ir[2] | jz | |
| Flow Elapsed Time | 43 | N/A | None | 13.990 ns | ir[5] | not1 | |
| Flow OS Summary | 44 | N/A | None | 13.625 ns | ir[3] | movc | |
| Flow Log | 45 | N/A | None | 13.585 ns | ir[2] | jc | |
| Analysis & Synthesis | 46 | N/A | None | 13.533 ns | ir[5] | in1 | |
| Fitter | 47 | N/A | None | 13.414 ns | ir[5] | add | |
| Assembler | 48 | N/A | None | 13.296 ns | ir[6] | sub | |
| Timing Analyzer | 49 | N/A | None | 13.202 ns | ir[2] | movc | |
| Summary | 50 | N/A | None | 12.869 ns | ir[6] | rsr | |
| Settings | 51 | N/A | None | 12.861 ns | ir[3] | movb | |
| Parallel Comp | 52 | N/A | None | 12.831 ns | en | sub | |
| tpd | 53 | N/A | None | 12.697 ns | ir[6] | nop | |
| Messages | 54 | N/A | None | 12.661 ns | ir[4] | and1 | |
| | 55 | N/A | None | 12.657 ns | ir[5] | halt | |
| | 56 | N/A | None | 12.648 ns | ir[5] | movc | |
| | 57 | N/A | None | 12.635 ns | ir[7] | rsr | |
| | 58 | N/A | None | 12.571 ns | en | rsr | |
| | 59 | N/A | None | 12.563 ns | ir[7] | jmp | |
| | 60 | N/A | None | 12.455 ns | ir[7] | jz | |
| | 61 | N/A | None | 12.438 ns | ir[2] | movb | |
| | 62 | N/A | None | 12.405 ns | en | nop | |
| | 63 | N/A | None | 12.243 ns | ir[7] | add | |
| | 64 | N/A | None | 12.183 ns | ir[5] | movb | |
| | 65 | N/A | None | 12.129 ns | ir[5] | out1 | |
| | 66 | N/A | None | 11.970 ns | ir[7] | jc | |
| | 67 | N/A | None | 11.545 ns | ir[7] | in1 | |
| | 68 | N/A | None | 11.485 ns | ir[7] | halt | |
| | 69 | N/A | None | 11.063 ns | ir[7] | movc | |
| | 70 | N/A | None | 10.784 ns | ir[4] | jmp | |
| | 71 | N/A | None | 10.784 ns | ir[4] | sub | |
| | 72 | N/A | None | 10.676 ns | ir[4] | jz | |
| | 73 | N/A | None | 10.671 ns | ir[1] | movc | |
| | 74 | N/A | None | 10.382 ns | ir[4] | movc | |
| | 75 | N/A | None | 10.344 ns | ir[7] | and1 | |
| | 76 | N/A | None | 10.261 ns | ir[4] | rsr | |
| | 77 | N/A | None | 10.204 ns | ir[4] | nop | |
| | 78 | N/A | None | 10.191 ns | ir[4] | ic | |

| Compilation Report | tpd | | | | | |
|----------------------|-----|-------|-------------------|-----------------|-------|------|
| Legal Notice | | Slack | Required P2P Time | Actual P2P Time | From | To |
| Flow Summary | 69 | N/A | None | 11.063 ns | ir[7] | movc |
| Flow Settings | 70 | N/A | None | 10.784 ns | ir[4] | jmp |
| Flow Non-Default | 71 | N/A | None | 10.784 ns | ir[4] | sub |
| Flow Elapsed Time | 72 | N/A | None | 10.676 ns | ir[4] | jz |
| Flow OS Summary | 73 | N/A | None | 10.671 ns | ir[1] | movc |
| Flow Log | 74 | N/A | None | 10.382 ns | ir[4] | movc |
| Analysis & Synthesis | 75 | N/A | None | 10.344 ns | ir[7] | and1 |
| Fitter | 76 | N/A | None | 10.261 ns | ir[4] | rsl |
| Assembler | 77 | N/A | None | 10.204 ns | ir[4] | nop |
| Timing Analyzer | 78 | N/A | None | 10.191 ns | ir[4] | jc |
| Summary | 79 | N/A | None | 10.153 ns | ir[4] | rsr |
| Settings | 80 | N/A | None | 10.048 ns | ir[7] | not1 |
| Parallel Comp | 81 | N/A | None | 9.725 ns | ir[4] | not1 |
| tpd | 82 | N/A | None | 9.511 ns | ir[0] | movc |
| Messages | 83 | N/A | None | 9.430 ns | ir[0] | rsr |
| | 84 | N/A | None | 9.205 ns | ir[1] | rsr |
| | 85 | N/A | None | 8.938 ns | ir[4] | movc |
| | 86 | N/A | None | 8.832 ns | ir[4] | in1 |
| | 87 | N/A | None | 8.821 ns | ir[4] | add |
| | 88 | N/A | None | 8.671 ns | ir[1] | movc |
| | 89 | N/A | None | 8.443 ns | ir[0] | movc |
| | 90 | N/A | None | 8.184 ns | ir[7] | out1 |
| | 91 | N/A | None | 8.171 ns | ir[7] | sub |
| | 92 | N/A | None | 8.158 ns | ir[0] | jmp |
| | 93 | N/A | None | 8.066 ns | ir[4] | halt |
| | 94 | N/A | None | 8.047 ns | ir[0] | jz |
| | 95 | N/A | None | 7.940 ns | ir[7] | movc |
| | 96 | N/A | None | 7.921 ns | ir[4] | movb |
| | 97 | N/A | None | 7.867 ns | ir[1] | jmp |
| | 98 | N/A | None | 7.866 ns | ir[4] | out1 |
| | 99 | N/A | None | 7.781 ns | ir[7] | nop |
| | 100 | N/A | None | 7.765 ns | ir[1] | jz |
| | 101 | N/A | None | 7.749 ns | ir[7] | movb |
| | 102 | N/A | None | 7.625 ns | ir[7] | rsl |
| | 103 | N/A | None | 7.562 ns | ir[0] | jc |
| | 104 | N/A | None | 7.350 ns | ir[1] | jc |
| | 105 | N/A | None | 7.189 ns | ir[0] | rsl |
| | 106 | N/A | None | 6.989 ns | ir[1] | rsl |

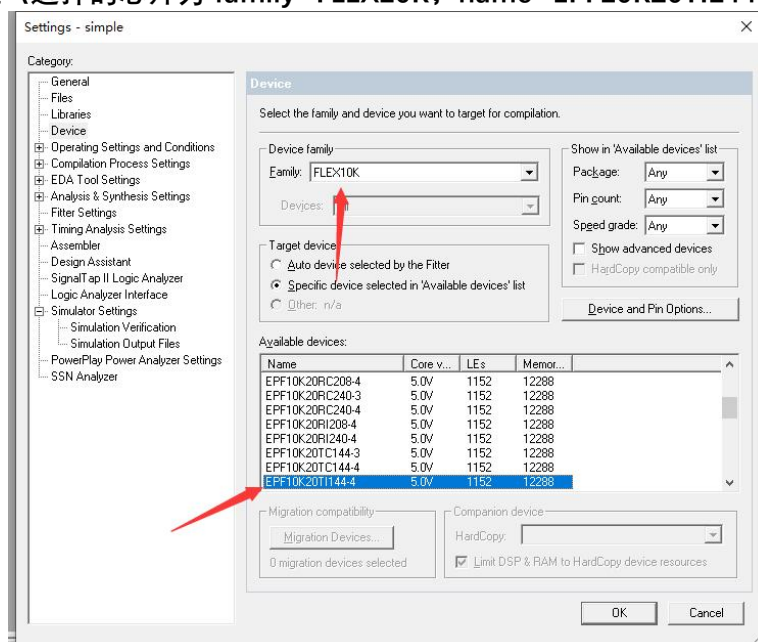
结果分析及结论：

Summary 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。

不同的端口由于路径不同，延时时间的也不同。延时范围为 6.989~18.116ns,且集中在 12-14ns 之间。延时分布集中有利于减轻冒险导致的毛刺时长，让有效的输出占比更大。并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

```

1 module simple(
2     input m,
3     input [3:0] s,
4     input [7:0] a,b,
5     output reg cf,zf,
6     output reg [7:0] t);
7     reg [8:0] temp;
8     always @(m or s or a or b) begin
9         zf=0;cf=0;
10        if(m==1'b0 && s[3:0]==4'b1010) t=b;
11        else if(m==1'b0 && (s[3:0]==4'b1100 || s[3:0]==4'b0100)) t=a;
12        else if(m==1'b1) begin
13            if(s[3:0]==4'b1001) begin
14                t=a+b;
15                if(t==8'b00000000) zf=1;
16                else zf=0;
17                temp=a+b;
18                if(temp[8]==1'b0) cf=0;
19                else cf=1;
20            end
21            else if(s[3:0]==4'b0110) begin
22                t=b-a;
23                if(t==8'b00000000) zf=1;
24                else zf=0;
25                temp=b-a;
26                if(temp[8]==1'b0) cf=0;
27                else cf=1;
28            end
29            else if(s[3:0]==4'b1011) t=a&b;
30            else if(s[3:0]==4'b0101) t=~b;
31        end
32    end
33 end
34 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 (10240): Verilog HDL Always Construct warning at simple.v(8): inferring latch(es) for variable "t", which holds its previous value in one or more paths through the always construct | |
| Warning (10240): Verilog HDL Always Construct warning at simple.v(8): inferring latch(es) for variable "temp", which holds its previous value in one or more paths through the always construct | |
| Warning: Latch t[0] latch has unsafe behavior | |
| Warning: Latch t[1] latch has unsafe behavior | |
| Warning: Latch t[2] latch has unsafe behavior | |
| Warning: Latch t[3] latch has unsafe behavior | |
| Warning: Latch t[4] latch has unsafe behavior | |
| Warning: Latch t[5] latch has unsafe behavior | |
| Warning: Latch t[6] latch has unsafe behavior | |
| Warning: Latch t[7] latch has unsafe behavior | |
| Warning: Timing Analysis is analyzing one or more combinational loops as latches | |
| Warning: Found pins functioning as undefined clocks and/or memory enables | |
| Warning: Found 8 node(s) in clock paths which may be acting as ripple and/or gated clocks -- node(s) analyzed as buffer(s) resulting in clock skew | |

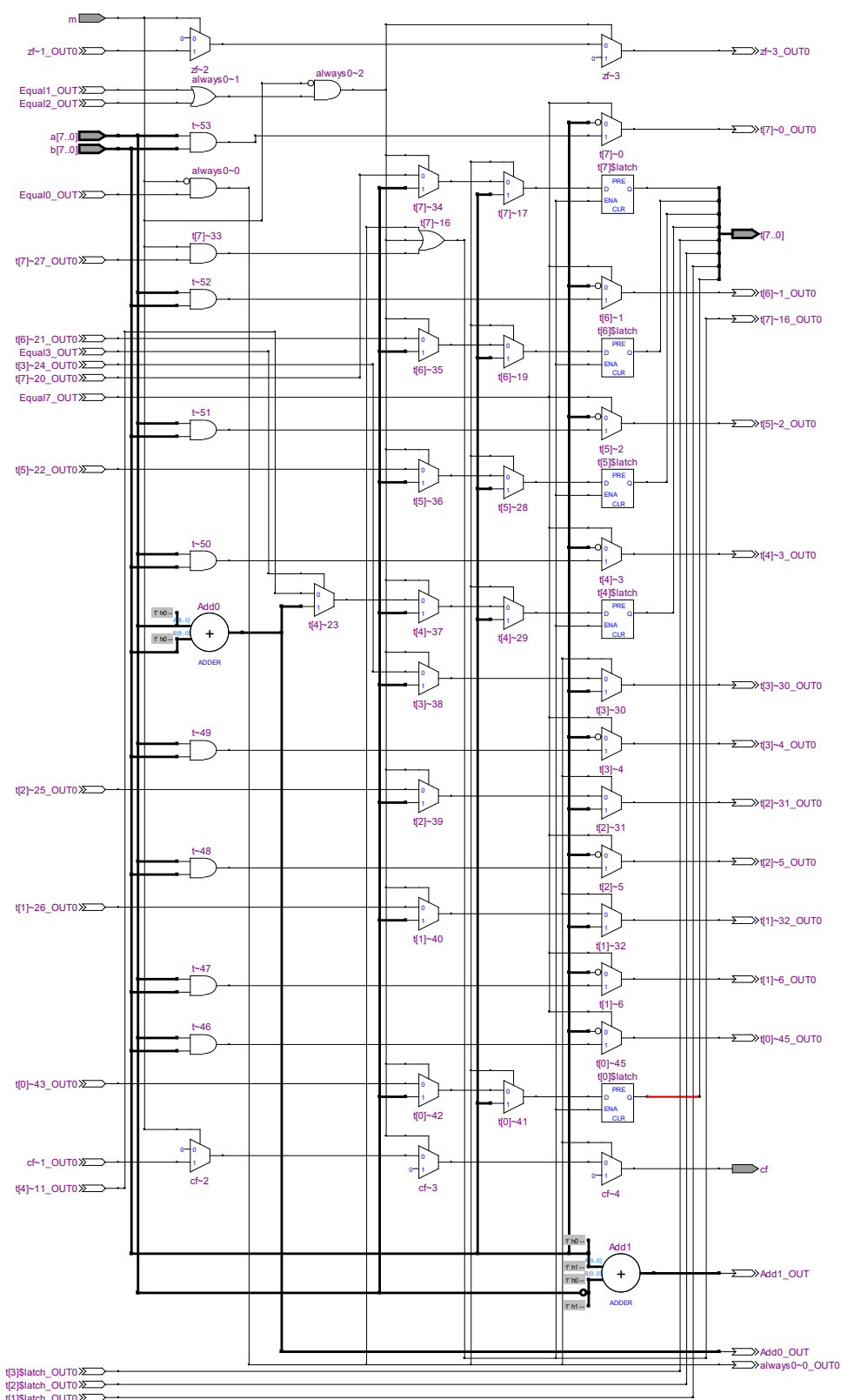
资源消耗：

```

Fitter Status           Successful - Thu Nov 17 00:35:56 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
Total logic elements     76 / 1,152 ( 7 % )
Total pins               31 / 102 ( 30 % )
Total memory bits        0 / 12,288 ( 0 % )

```

D) RTL 视图



结果分析及结论：

代码出现提示：Warning: Timing Analysis is analyzing one or more combinational loops as latches，需用异步加载数据信号的寄存器实现这些锁存器，或

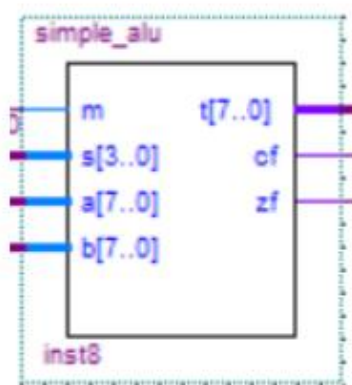
者索性从设计中删除他们。

代码出现提示：Warning: Found pins functioning as undefined clocks and/or memory enables，作为时钟的 PIN 没有约束信息。

由图可知，不存在锁存器，因此符合要求。一个功能的实现需要经过多重门的处理后才能实现，一个元件的内部原理结构图十分复杂。

电路运用加法器实现了加运算，还有大量选择器选择对应的结果输出，还有比较器。输入信号为 m, s, a, b, 输出信号为 t, cf, zf。

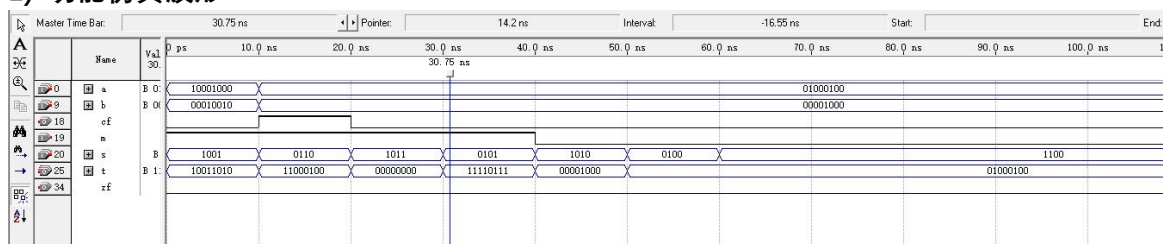
接口设计：



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：

| 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=/b(注: b 相反) | 不影响 | 不影响 |
| 0 | 1010 | t=b | 不影响 | 不影响 |
| 0 | 1100 或 0100 | t=a | 不影响 | 不影响 |

E) 功能仿真波形



结果分析及结论：

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 $t=\sim b$

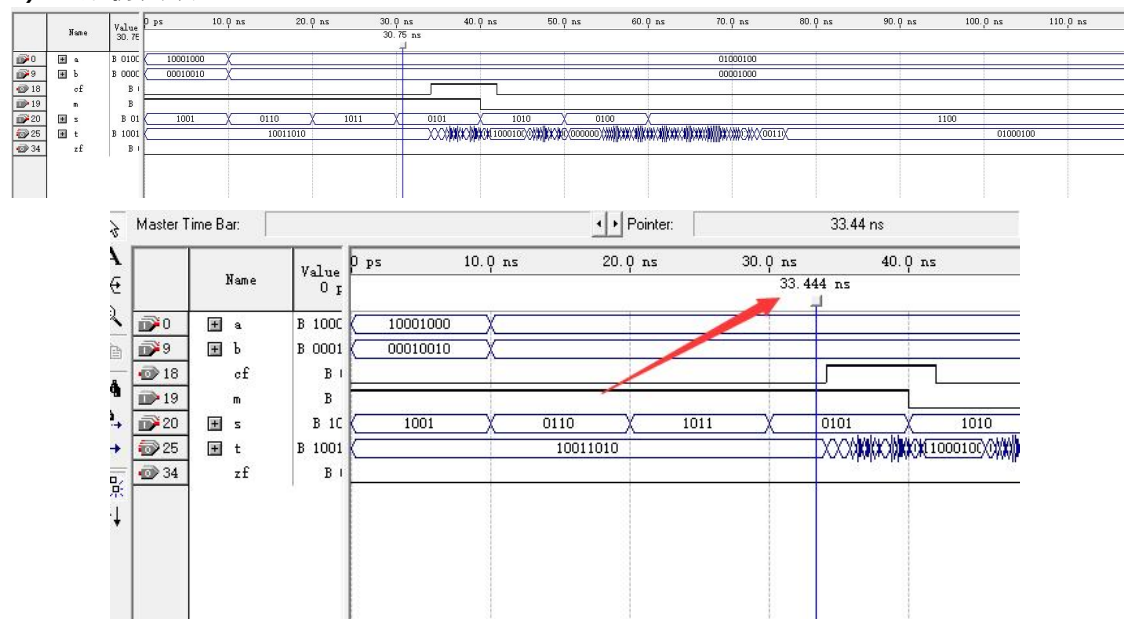
当控制信号 m 为 0, s 为 1010 时, 执行 $t=b$

当控制信号 m 为 0, s 为 1100 或 0100 时, 执行 $t=a$

当操作不影响 cf 和 zf 时, 为了避免产生锁存器, 赋初值为 0, cf 和 zf 输出 0。

由于存储 cf 和 zf 的寄存器的使能信号处于非使能状态, 因此输出的 cf 和 zf 不会被读入进寄存器。

F) 时序仿真波形



结果分析及结论：

在输入改变的时候, 输出会存在一段噪声, 随后输出正确的值。不同的操作对应的噪声时长也不同。

时序仿真输出的延时约 20ns, 且输出信号做出反应时产生了短时间内的信号波动。

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 tsu | N/A | None | 20.600 ns | b[0] | t[6]\$latch | -- | m | 0 |
| 2 | Worst-case tco | N/A | None | 31.000 ns | t[2]\$latch | t[2] | s[3] | -- | 0 |
| 3 | Worst-case tpd | N/A | None | 30.600 ns | b[0] | zf | -- | -- | 0 |
| 4 | Worst-case th | N/A | None | 12.000 ns | a[2] | t[2]\$latch | -- | s[3] | 0 |
| 5 | Total number of failed paths | | | | | | | | 0 |

| tpd | | | | | | |
|-----|-------|-------------------|-----------------|------|----|--|
| | Slack | Required P2P Time | Actual P2P Time | From | To | |
| 1 | N/A | None | 30.600 ns | b[0] | zf | |
| 2 | N/A | None | 30.200 ns | a[1] | zf | |
| 3 | N/A | None | 30.000 ns | b[1] | zf | |
| 4 | N/A | None | 29.800 ns | b[2] | zf | |
| 5 | N/A | None | 29.400 ns | a[3] | zf | |
| 6 | N/A | None | 29.400 ns | a[0] | zf | |
| 7 | N/A | None | 29.000 ns | a[2] | zf | |
| 8 | N/A | None | 28.700 ns | b[3] | zf | |
| 9 | N/A | None | 28.400 ns | b[4] | zf | |
| 10 | N/A | None | 27.800 ns | b[6] | zf | |
| 11 | N/A | None | 27.700 ns | a[4] | zf | |
| 12 | N/A | None | 27.400 ns | b[5] | zf | |
| 13 | N/A | None | 27.300 ns | a[5] | zf | |
| 14 | N/A | None | 27.100 ns | a[6] | zf | |
| 15 | N/A | None | 24.300 ns | b[0] | cf | |
| 16 | N/A | None | 24.200 ns | a[7] | zf | |
| 17 | N/A | None | 24.200 ns | b[7] | zf | |
| 18 | N/A | None | 24.100 ns | s[3] | cf | |
| 19 | N/A | None | 23.900 ns | a[1] | cf | |
| 20 | N/A | None | 23.700 ns | b[1] | cf | |
| 21 | N/A | None | 23.500 ns | b[2] | cf | |
| 22 | N/A | None | 23.100 ns | a[3] | cf | |
| 23 | N/A | None | 23.100 ns | a[0] | cf | |
| 24 | N/A | None | 23.000 ns | s[3] | zf | |
| 25 | N/A | None | 22.700 ns | a[2] | cf | |
| 26 | N/A | None | 22.500 ns | s[2] | cf | |
| 27 | N/A | None | 22.500 ns | s[1] | cf | |
| 28 | N/A | None | 22.400 ns | b[3] | cf | |
| 29 | N/A | None | 22.100 ns | b[4] | cf | |
| 30 | N/A | None | 22.000 ns | s[0] | cf | |
| 31 | N/A | None | 21.500 ns | b[6] | cf | |
| 32 | N/A | None | 21.400 ns | s[2] | zf | |
| 33 | N/A | None | 21.400 ns | s[1] | zf | |
| 34 | N/A | None | 21.400 ns | a[4] | cf | |
| 35 | N/A | None | 21.100 ns | b[5] | cf | |
| 36 | N/A | None | 21.000 ns | a[5] | cf | |
| 37 | N/A | None | 20.900 ns | s[0] | zf | |
| 38 | N/A | None | 20.800 ns | a[6] | cf | |
| 39 | N/A | None | 17.800 ns | a[7] | cf | |
| 40 | N/A | None | 17.800 ns | b[7] | cf | |
| 41 | N/A | None | 17.700 ns | m | cf | |
| 42 | N/A | None | 17.000 ns | m | zf | |

结果分析及结论：

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。

不同的端口由于路径不同，延时时间的也不同。延时范围为 17.000~30.600ns,且集中在 21-23ns 之间。延时分布集中有利于减轻冒险导致的毛刺时长，让有效的输出占比更大。并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？不必须。

控制信号对应指令编码如下：

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

例如移位控制信号产生逻辑：执行 MOV、ADD、SUB、AND、NOT、OUT 指令时，ALU 输出的数据需通过移位逻辑直传至总线 BUS。

比如：jmp 和 add 是可以合并的，因为 jmp 是将 add 后的结果写入 pc 中，可以进行 add 操作后直接进行写入操作。

比如：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？

ALU 的 S[3..0]来自指令码 ir 的前四位，与指令寄存器存储的指令前 4 位相同。

3、为何 S[3..0]等于 1100 时将输入 a 传给 t，S[3..0]等于 1010 或 0100 时将输入 b 传给 t？

当 S[3..0]为 1100 时，执行传输指令，判断源和目的寄存器，因为不需要写入数据，所以从 RAA 输入源寄存器；因为需要写入数据，所以 RWBA 输入目的寄存器。源寄存器中的数据从 S 口传出，经过 ALU 和移位逻辑，传送到总线上，最终输入目的端，然后给 t 赋值。控制输出 t 等于 a。

当 S[3..0]为 1010 时，是移位操作，因为源和目的都是同一个寄存器 B，故从 D 口传出数据，在移位逻辑对数据进行操作，传回目的寄存器 B，只能对 B 中数据进行操作。

当 S[3..0]为 0100 时, 是 out 操作, 因为 01, 所以目的寄存器是 B, 故从经过 ALU 和移位逻辑, 传输到总线上, 再输出, 给 t 赋值。

当 S[3..0]等于 1010 或 0100 时, t 等于 b, 此时 alu 相当于选择器。

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:

- ✧ quatus-||的操作与仿真;
- ✧ verilog 语言编写电路;
- ✧ 模型机的内部结构和工作原理;
- ✧ 指令译码器的原理;
- ✧ ALU 的工作原理;

解决的办法: 询问老师和同学+百度。

遇到的问题: 在编写模型机各个部件时没有全局观念, 没有考虑把这个部件最终组合后可能的运行情况, 使得设计会出现一定的差错。

经验教训: 要学会多尝试, 这样才能发现自己的问题。在写代码之前多研究实验指导。

2、对本实验内容、过程和方法的改进建议 (可选项)。