# 实验三 模型机组合部件的实现（二）

班级 信安2302班 姓名 石云博 学号 202308060227

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，2-1 多路复用器。
3. 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8重2-1 多路复用器；
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=Cyclone II；name=EP2C5T144C8）

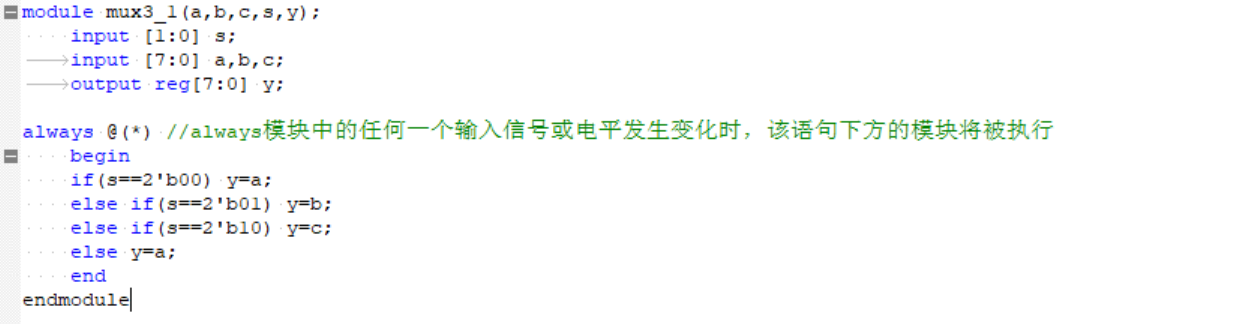
步骤：左上角 file->New Project Wizard->选择工程位置和工程名->选择芯片 Cyclone II，available device中选择 EP2C5T144C8->点击 next->最后点击 finish 完成创建工程

B) 编写源代码

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：

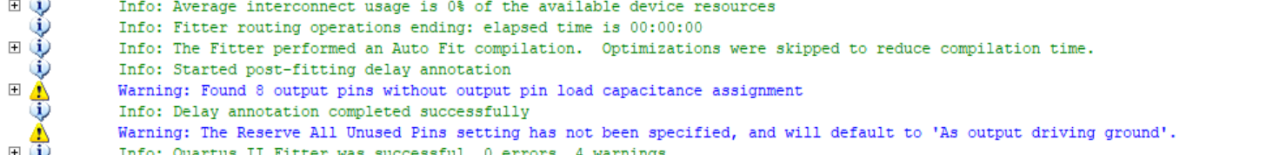


C) 编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）

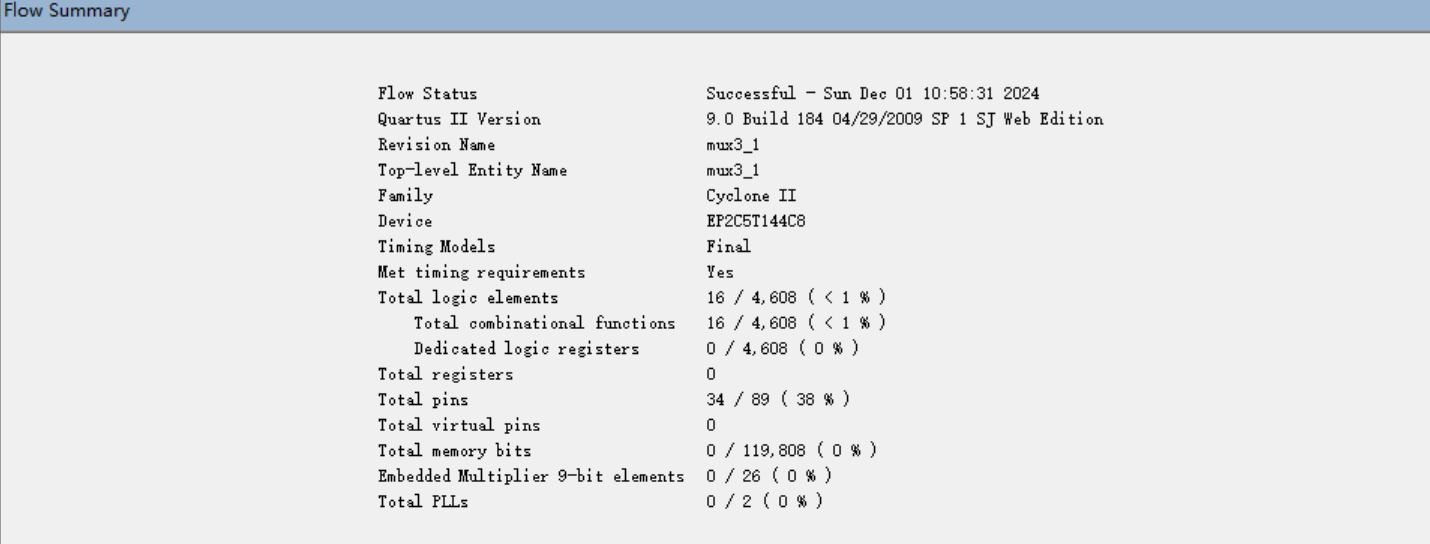
确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功，保存文件。

无错误。

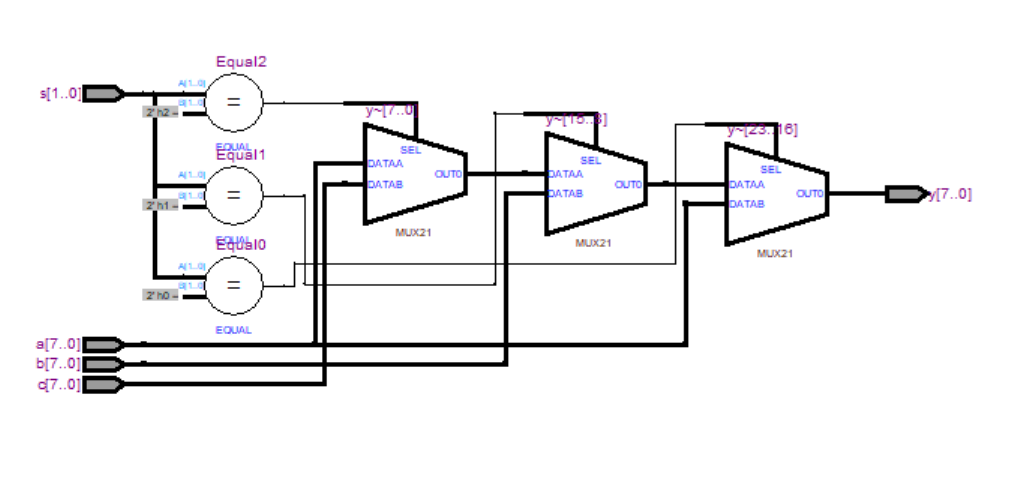
警告信息：



资源消耗：



D) RTL视图



视图分析及结论：

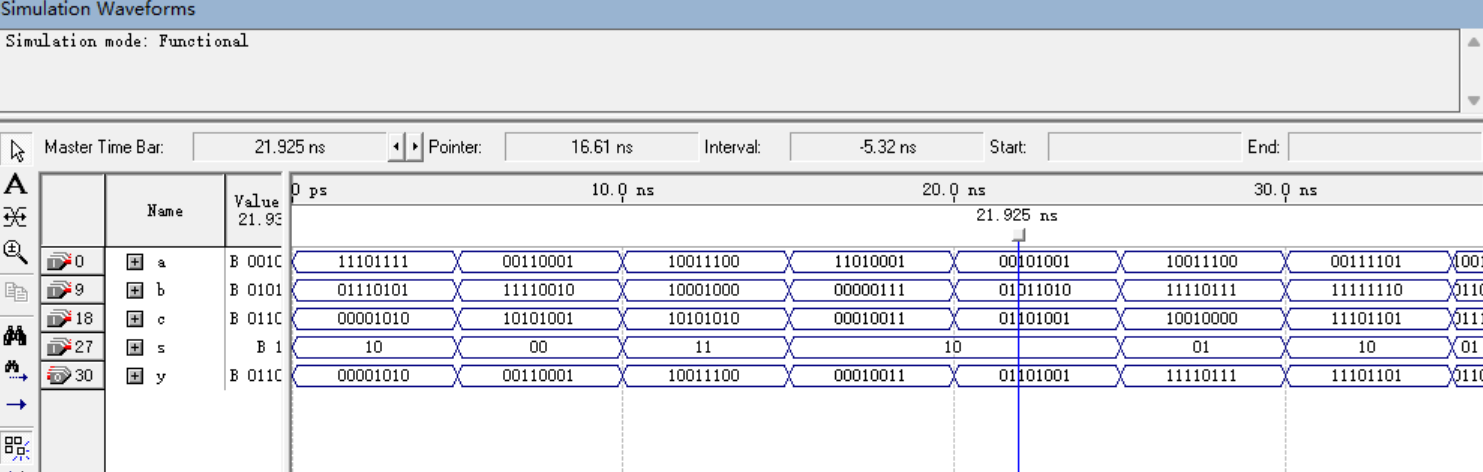
视图分析：视图左边为输入，右边为输出。其中连接有大量的元器件。例如比较器：当输入相等时输出1，不相等时输出0；还由大量的2-1选择器构成，当控制信号为0时，输出第一位，控制信号1时，输出第二位。图中输入信号为s和a、b、c，输出信号为y。各个输出端口之间通过导线相连。

结论：8重3-1多路复用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。

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；

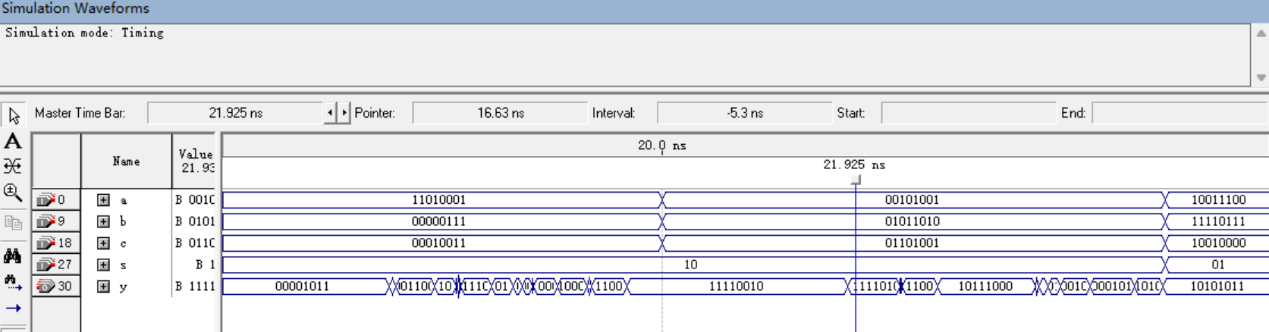
s=10 时，控制输出 y 等于 c，正确；s=11 时，控制输出高阻态，正确；

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：



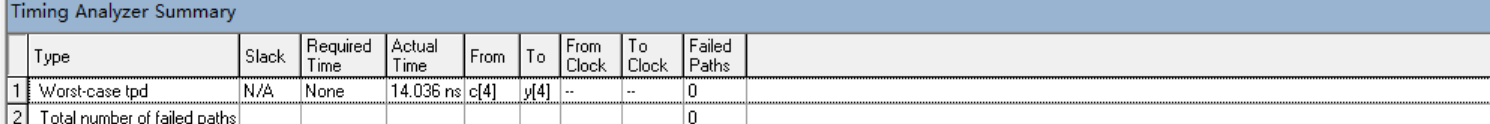
结果分析及结论：

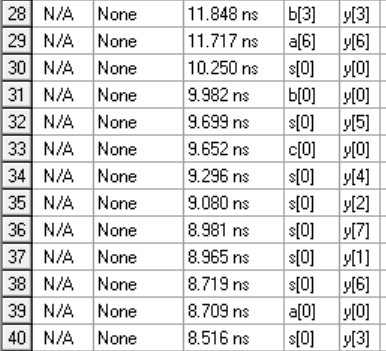
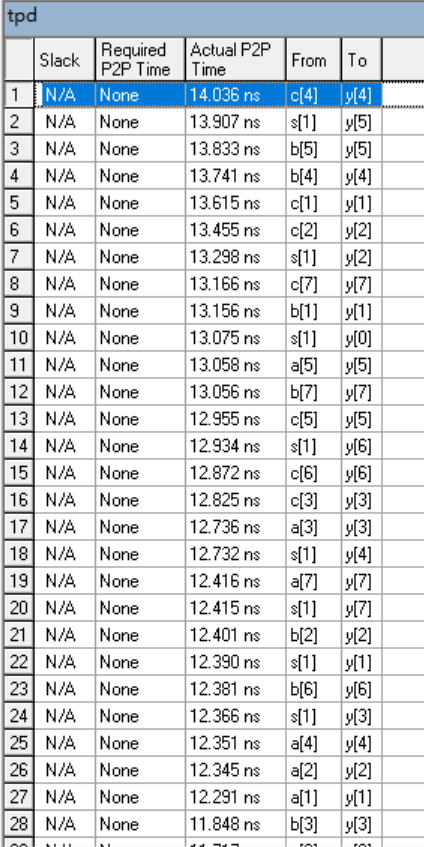
结果分析：时序仿真是指在布线后进行，是最接近真实器件运行的仿真，它与特定的器件有关，又包含了器件和布线的延时信息。由波形可得，当输入状态发生改变时，输出结果并未同时改变，而是有一定延迟，同时由于输入状态的改变，导致电路出现“冒险”，导致输出结果并未与预期结果相同。

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】





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c[4]到y[4]的最坏定时情况的tpd为14.036ns。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s[1]到y[7]的tpd 为14.907ns。

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、8重2-1多路复用器

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

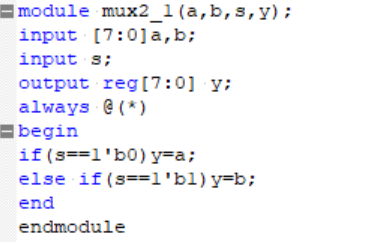
步骤：左上角 file->New Project Wizard->选择工程位置和工程名->选择芯片 **Cyclone II**，available device中选择 **EP2C5T144C8-**>点击 next->最后点击 finish 完成创建工程

B) 编写源代码

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：

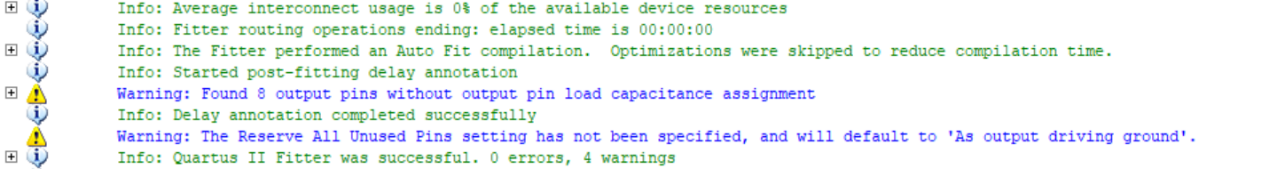


C) 编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）

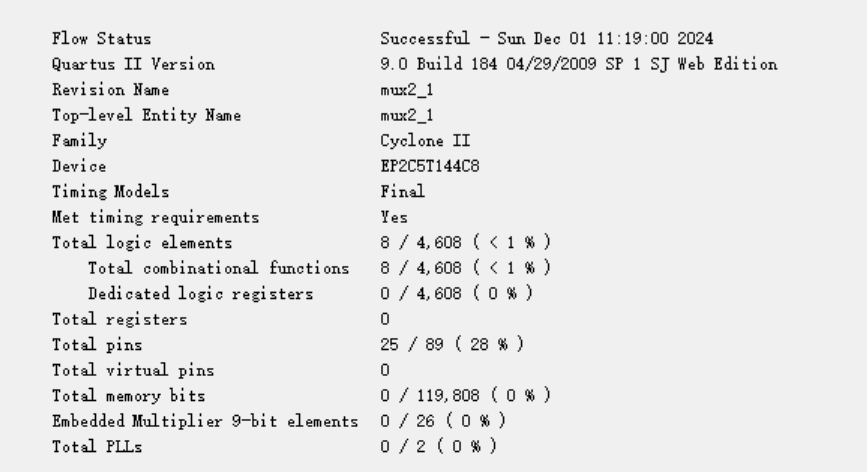
确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功，保存文件。

无错误。

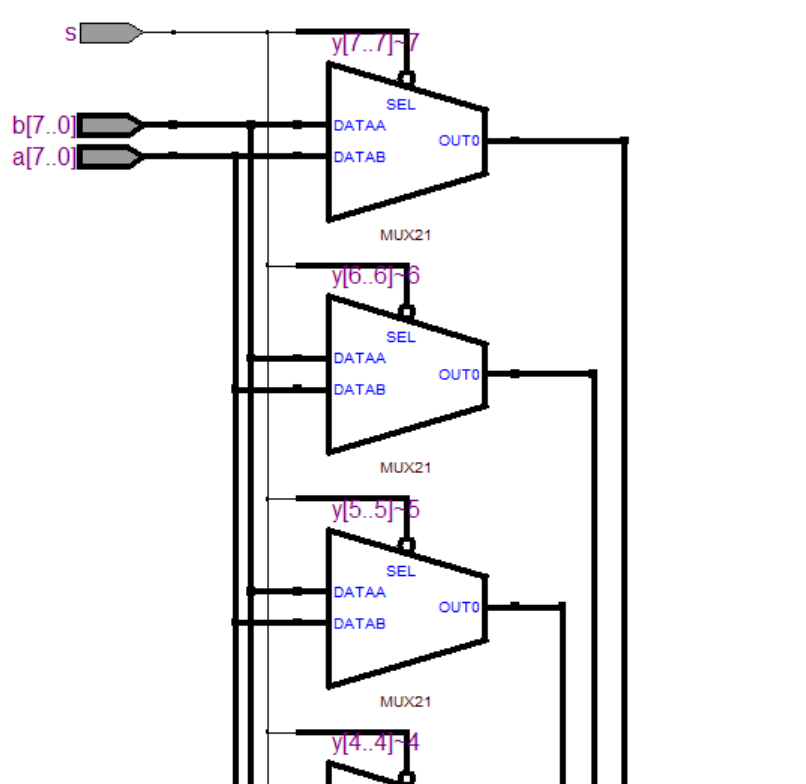
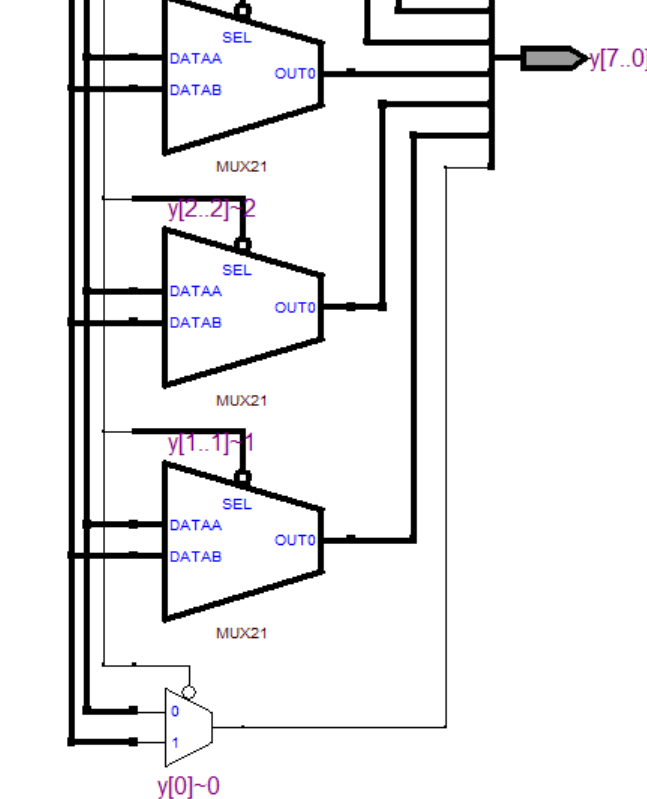
警告信息：



资源消耗：



D) RTL视图

视图分析及结论：

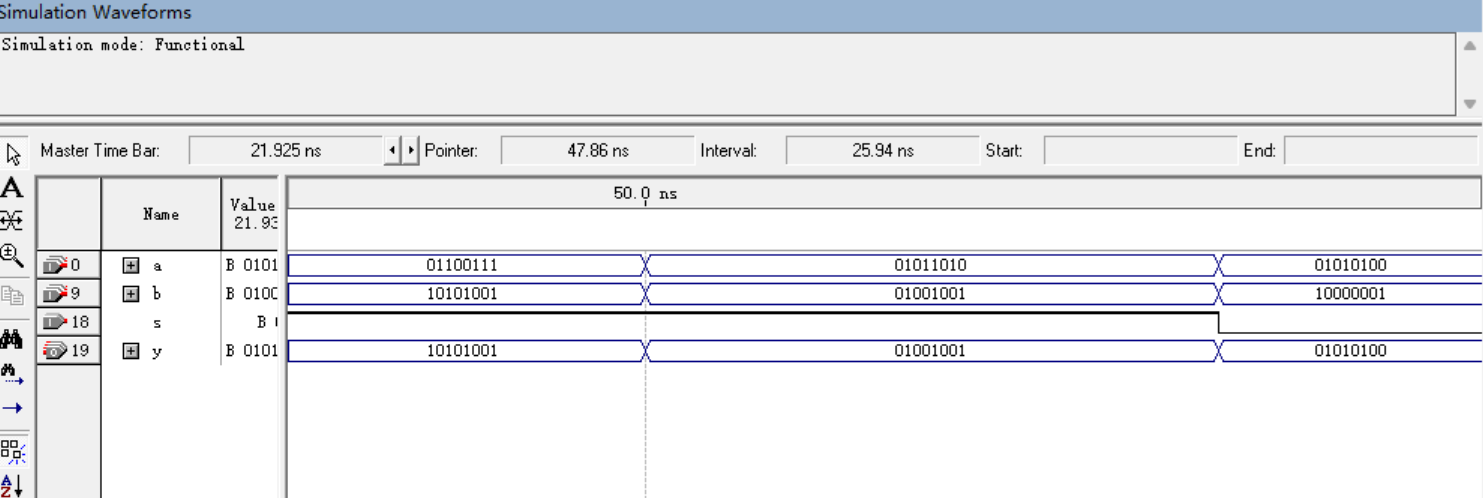
视图分析：通过观察 RTL 视图可知：左侧为输入，右侧为输入，电路中有很多的元器件进行连接，如：含有多路复用器，输入信号包括控制信号 s 以及 8 位数字信号 a、b，输出信号为 s 对 a、b 信号的选择，各个输出端口以及输入端口由导线相连接。

结论：移位逻辑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 s=0 时，输出 t=a;

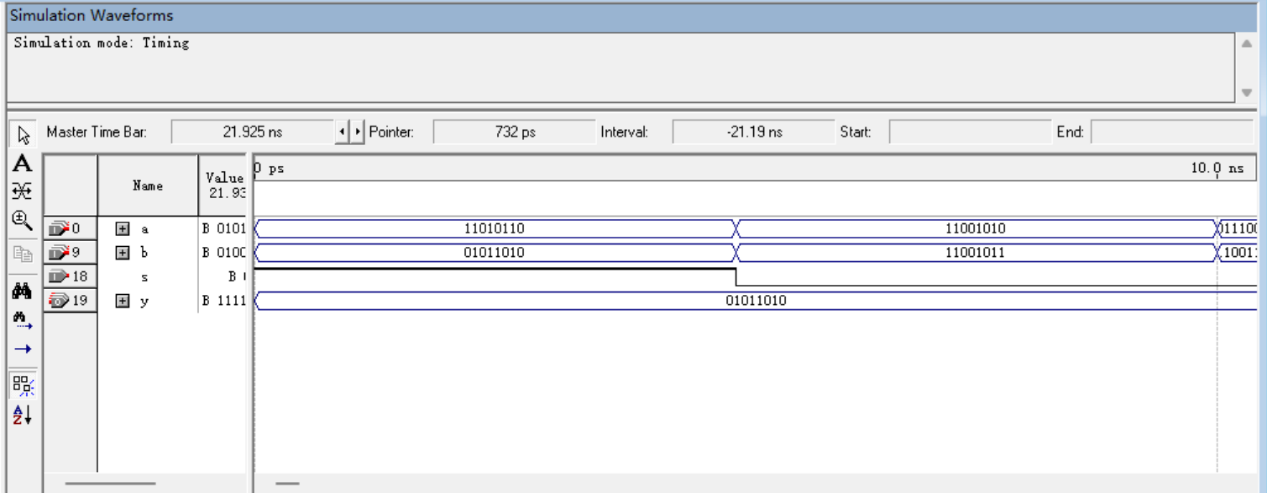
当 s=1 时，输出 y=b；

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：



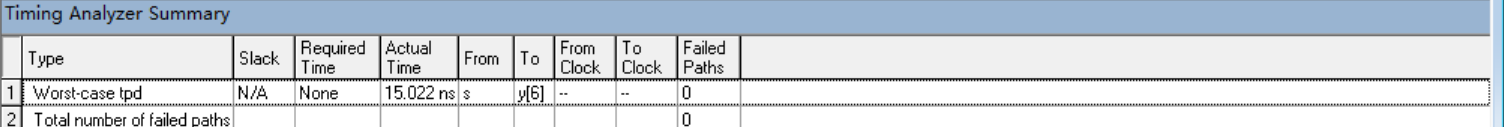
结果分析及结论：

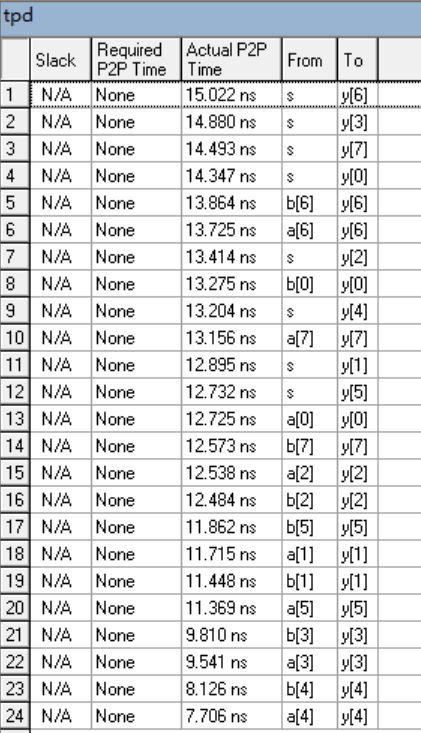
结果分析：时序仿真是指在布线后进行，是最接近真实器件运行的仿真，它与特定的器件有关，又包含了器件和布线的延时信息。由波形可得，当输入状态发生改变时，输出结果并未同时改变，而是有一定延迟，同时由于输入状态的改变，导致电路出现“冒险”，导致输出结果并未与预期结果相同。

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】





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s到y[6]的最坏定时情况的tpd为15.022ns。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，比如第三行中s到y[7]的tpd为14.439ns。

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。

3、控制信号产生逻辑

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

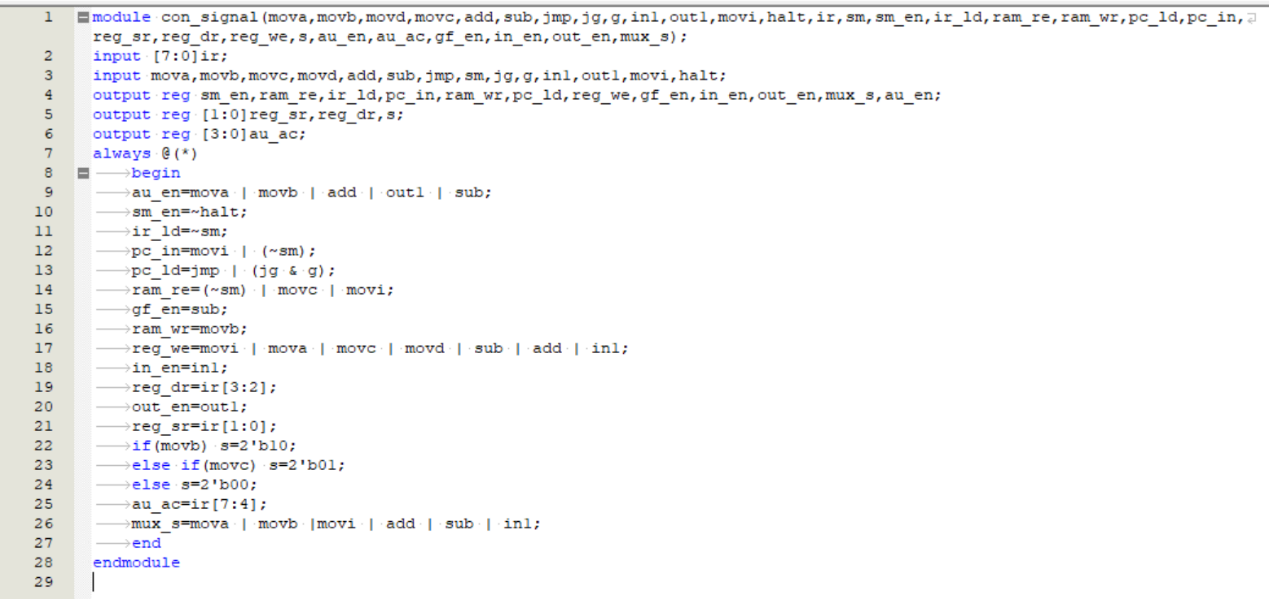
步骤：左上角 file->New Project Wizard->选择工程位置和工程名->选择芯片 **Cyclone II**，available device中选择 **EP2C5T144C8**->点击 next->最后点击 finish 完成创建工程

B) 编写源代码

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：

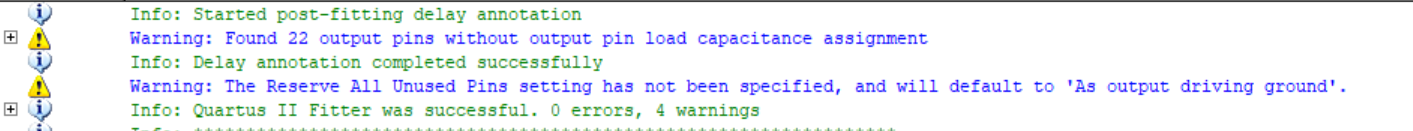


C) 编译与调试（包含编译调试过程中的错误、警告信息以及资源消耗）

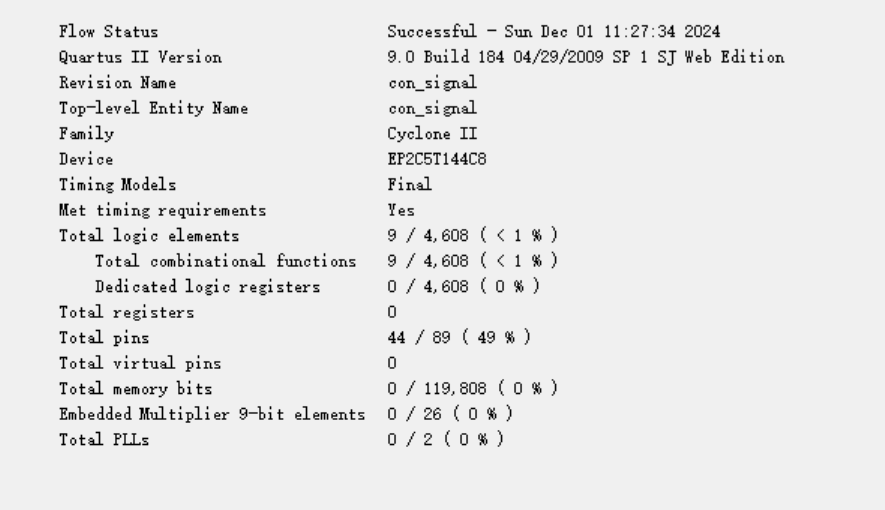
确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功，保存文件。

无错误。

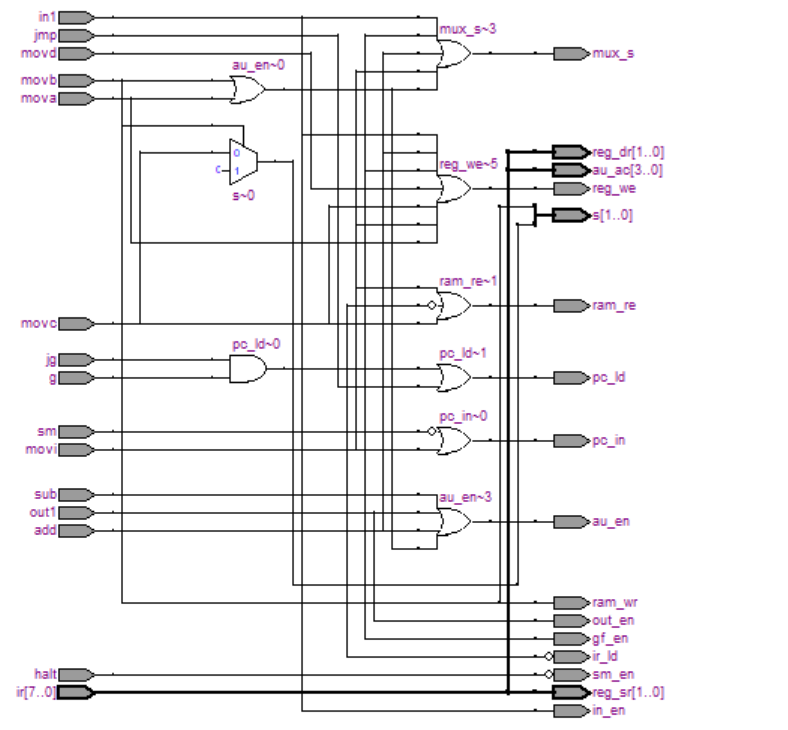
警告信息：



资源消耗：



D) RTL视图



视图分析及结论：

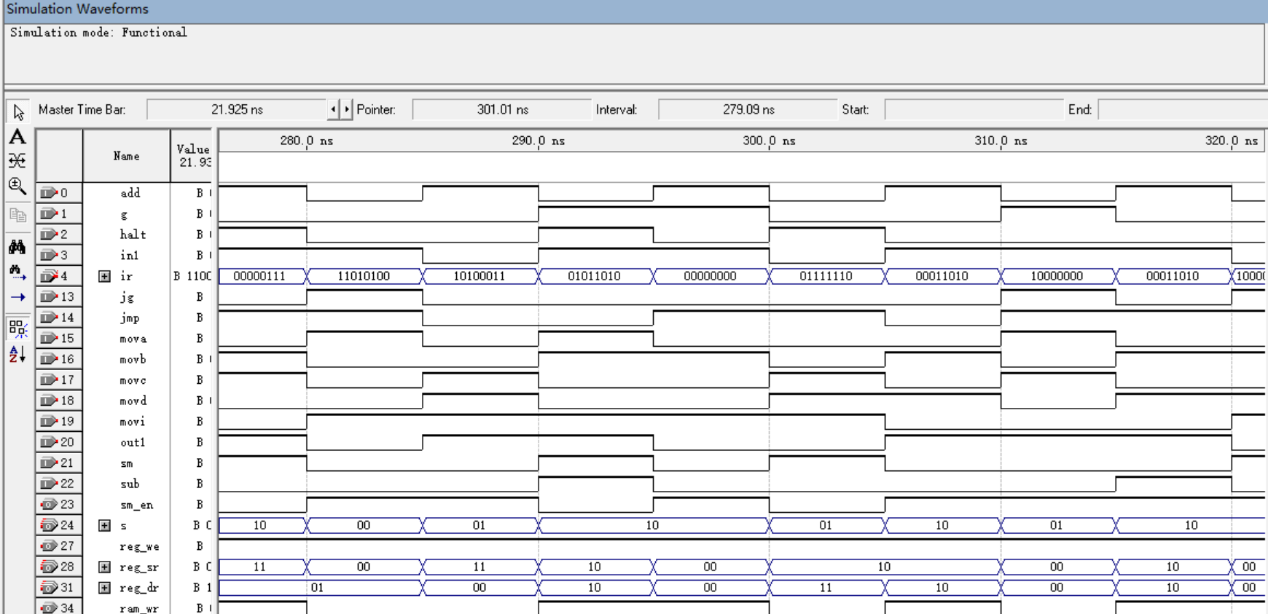
视图分析：视图左边为输入，右边为输出。其中连接有大量的元器件。例如比较器：当输入相等时输出1，不相等时输出0；还由大量的2-1选择器构成，当控制信号为0时，输出第一位，控制信号1时，输出第二位。图中输入信号为sm等15个输入，输出信号包括reg\_ra等16种情况。各个输出端口之间通过导线相连。

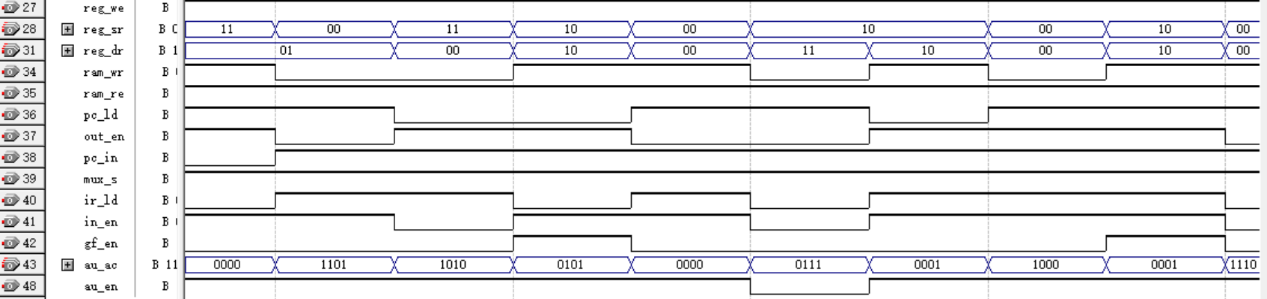
结论：控制信号产生逻辑功能需要经过多重门的处理后才能实现，其内部原理结构图十分复杂。

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。

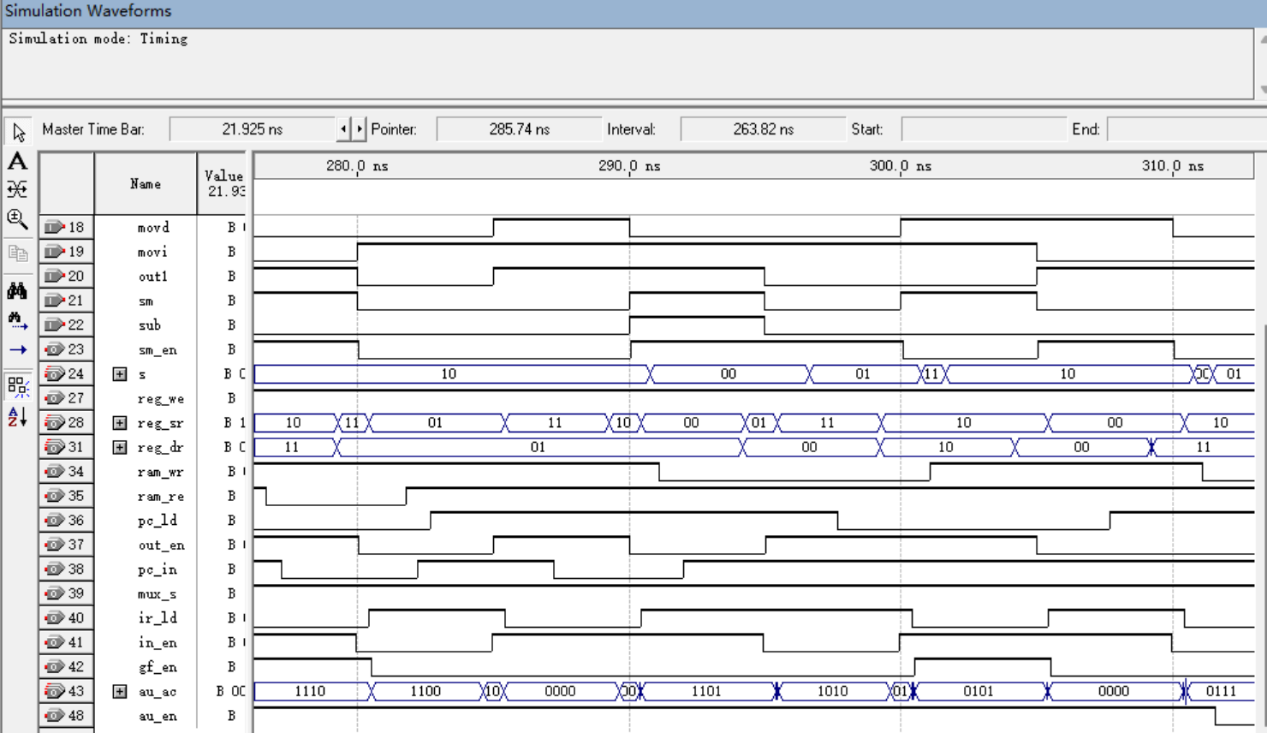
1. 指令码 ir 为 10100111 时，au\_ac=1010，reg\_dr=01，reg\_sr=11；
2. mova 指令执行时，sm\_en 输出为 1，au\_en 输出为 1，mux\_s 输出为 1，reg\_we 输出为 1；
3. movb 指令执行时，sm\_en 输出为 1，ram\_wr 输出为 1，au\_en 输出为 1，mux\_s 输出为 1；
4. movc 指令执行时，sm\_en 输出为 1，ram\_re 输出为 1，reg\_we 输出为 1；
5. movd 指令执行时，sm\_en 输出为 1，reg\_we 输出为 1；
6. add 指令执行时，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。

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：



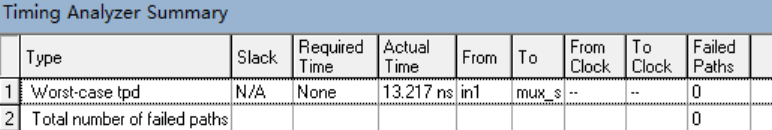
结果分析及结论：

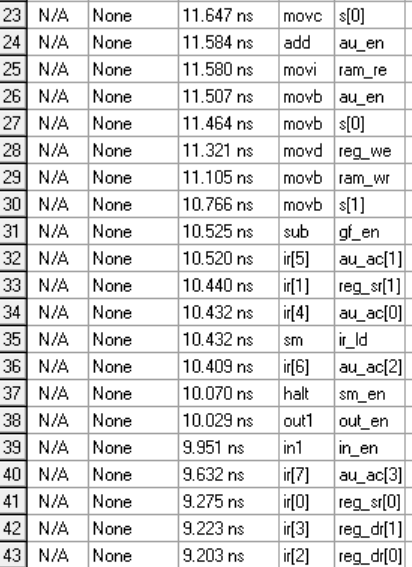
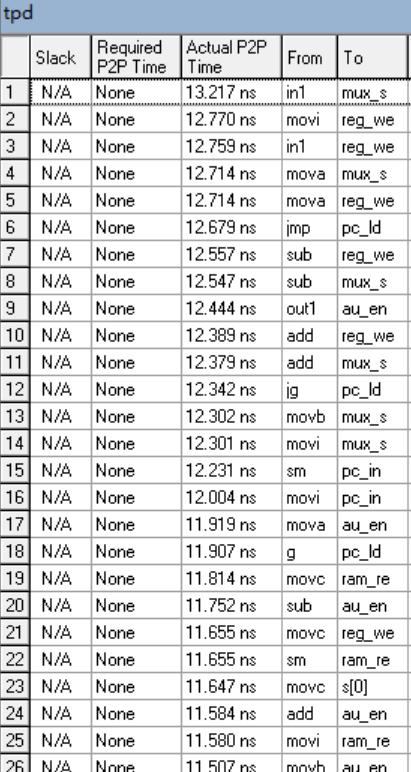
结果分析：时序仿真是指在布线后进行，是最接近真实器件运行的仿真，它与特定的器件有关，又包含了器件和布线的延时信息。由波形可得，当输入状态发生改变时，输出结果并未同时改变，而是有一定延迟，同时由于输入状态的改变，导致电路出现“冒险”，导致输出结果并未与预期结果相同。

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】





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in1到mux\_s的最坏定时情况的tpd为13.217ns。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movi到reg\_we的tpd 为12.770ns。

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、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。

答：选择 mova 指令，根据控制信号 SR1、SR0（即指令码 ir 的最后两位）将源寄存器 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，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。执行时sm\_en、au\_en、mux\_s、reg\_we 为 1，其他信号为 0。

2．如何产生正确的控制信号以及具体的编程实现？

答：应当逐个分析每个控制信号在不同的指令下对应的状态，利用逻辑函数进行状态的总和,分析得出最后的表达式，从而在 Quartus 中使用 Verilog 语言实现。

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。

答：这次实验我基本掌握了8 重 3-1 多路复用器，8 重 2-1 多路复用器，控制信号产生逻辑的工作原理及如何构造，让我对模型机的工作原理有了更加深入的了解，同时在运行调试过程中的锁存器问题也让我对组合电路以及时序电路的区别与表现理解更为细致。本次实验的核心在于理解CPU中命令的执行情况和各种控制信号对应的输出变化。在编程实现时虽然实现语句很简单，但需要非常细致，对每一步命令执行的输入输出非常了解，不然很容易在逻辑上出现错误。在编写模型机各个部件时也应当具有全局观念，要考虑把这个部件最终组合后可能的运行情况，使得设计的差错尽可能在最初的时候就能避免。Verilog程序调试时会经常遇到各种各样的问题，但通过上网查询相关资料和询问同学后得以解决问题。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）。

答：最好能给出控制信号产生逻辑的各个控制信号的输出功能表