计算机系统结构课程实验
总结报告

实验题目：动、静态流水线设计与性能对比分析

学号：1553534

姓名：　李帅

指导教师：　秦国峰

日期：2017/12/31

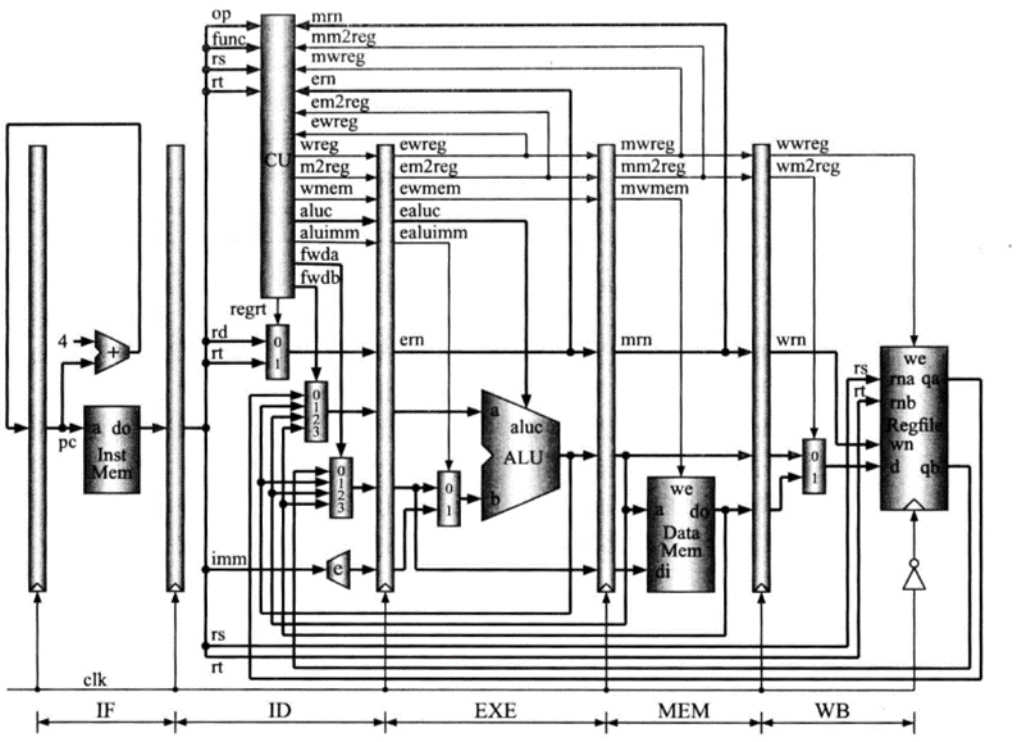
一、 实验环境部署与硬件配置说明

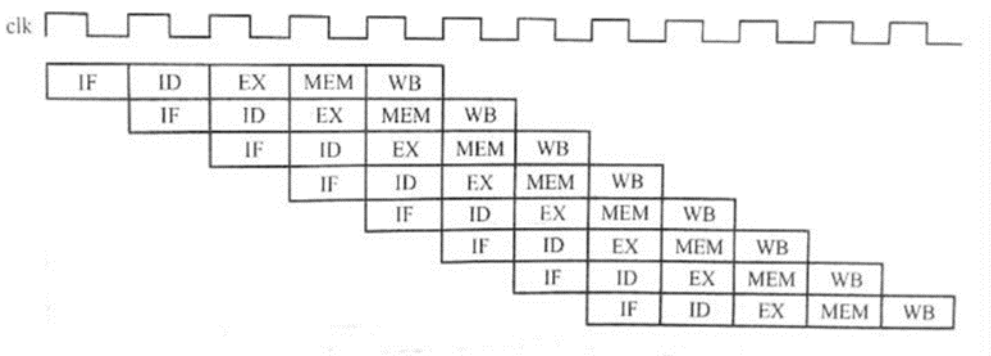
Win10 系统，Vivado2016，N4 板

二、 实验的总体结构

1、 静态流水线的总体结构

根据 MIPS 处理器指令的特点，将整体的处理过程分为取指令（IF）、指令译码（ID）、执行（EX）、 存储器访问（MEM）和寄存器回写（WB）五级，对应多周期 CPU 的五个处理阶段。一个指令的执行需 要 5 个始终周期，每个时钟周期的上升沿来临时，此指令所代表的一系列数据和控制信息将转移到下一级 处理，由于静态流水线在同一时间内只能各段按照同一种功能的连接方式工作。故设计气泡和计时器暂停解决冲突，计算应该暂停的周期然后使用相关计数器进行暂停操作。





2、 动态流水线的总体结构

动态流水线在同一时间内可，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能，允许在某些段中实现某种运算却在另外一些段中实现另外一种功能，因而更加灵活。

总体设计与静态流水线大部分类似，关于冲突，在 ex 模块中加入定向技术（forwarding）

三、 总体架构部件的解释说明

1、 静态流水线总体结构部件的解释说明

本实验分为 5 级流水线，分别是 IF,ID,EXE,MEM,WB，（命名方式按照《计算机原理与设计》规范）其中

IF 级：取指令级。从 ROM（ip 核，装入 coe 文件） 中读取指令，并在下一个时钟沿到来时把指令送到 ID 级的指令缓冲 器中。该级控制信号决定下一个指令指针的 PCS 信号、阻塞流水线 PC_IFwrite 信号、清空流水线的 IF_flush 信号；

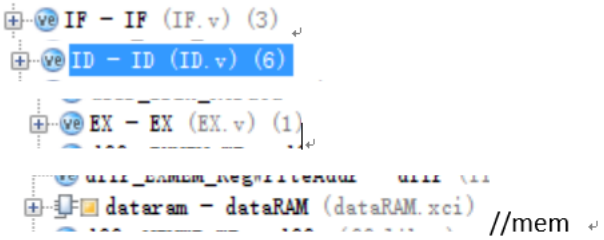
ID 级：指令译码级。对 IF 级来的指令进行译码，并产生相应的控制信号。整个 CPU 的控制信 号基本都是在这级上产生。该级自身不需要任何控制信号。并且，冒险检测也在该级，并计算暂停周期数。

EX 级：执行级。此级进行算数或逻辑操作。此外 LW、SW 指令所用的 RAM 访问地址也是在 本级上实现。控制信号有 ALUCode、ALUSrcA、ALUSrcB 和 RegDst，根据这些信号确定 ALU 操作、选 择两个 ALU 操作数 A、B，并确定目标寄存器。

MEM 级：存储器访问级。只有在执行 LW、SW 指令时才对存储器进行读写，对其他指令只起 到缓冲一个周期的作用。该级只需要存储器写操作允许信号 MemWrite

WB 级：回写级。此级把指令执行的结果回写到寄存器文件中。该级设置信号 MemToReg 和寄 存器写操作允许信号 RegWrite，其中 MemToReg 决定写入寄存器的数据来自于 MEM 级上的缓冲值或来 自于 MEM 级上的存储器。关于冲突，书上有三种冲突，分别是：结构冒险（mips 指令不存在此种风险）；数据冲突（用阻塞解决）；控制冲突（用分支延迟解决）

在实际代码中，建立了以下四个 module:



```
    output timeout;
//  reg x;
    assign stall=MemRead_ex&&((RegWriteAddr_ex==RsAddr_id)||(RegWriteAddr_e
    assign PC_IFWrite=~stall;
    assign timeout=(stall==1)?3:0;
```

在顶层模块中，assign 语句直接写出 wb 级模块，其中各级之间的流水寄存器均在顶层模块中实现。

顶层模块为 MipsPipelineCPU（后仿真以及下板）

在前仿真时，顶层模块为 top_tb；

每段流水线中的操作，完全参考课本

2、 动态态流水线总体结构部件的解释说明

（1）IF 级：取指令级。从 ROM 中读取指令，并在下一个时钟沿到来时把指令送到 ID 级的指令缓冲 器中。该级控制信号决定下一个指令指针的 PCSource 信号、阻塞流水线 PC_IFwrite 信号、清空流水线的 IF_flush 信号。

（2）ID 级：指令译码级。对 IF 级来的指令进行译码，并产生相应的控制信号。整个 CPU 的控制信 号基本都是在这级上产生。该级自身不需要任何控制信号。 流水线冒险检测也在该级上进行，冒险检测电路需要上一条指令的 MemRead，即在检测到毛细那条 件成立时，冒险检测电路会产生 stall 信号清空 ID/EX 寄存器，插入一个流水线气泡。

（3）EX 级：执行级。此级进行算数或逻辑操作。此外 LW、SW 指令所用的 RAM 访问地址也是在 本级上实现。控制信号有 ALUCode、ALUSrcA、ALUSrcB 和 RegDst，根据这些信号确定 ALU 操作、选 择两个 ALU 操作数 A、B，并确定目标寄存器。另外，数据转发也在该级完成。数据转发控制电路产生 FowardA 和 FowardB 两组控制信号。

```
//forwarding
    wire[1:0] ForwardA,ForwardB;

    assign ForwardA[0]=RegWrite_wb&&(RegWriteAddr_wb!=0)&&(RegWriteAddr_mem!=RsAddr_ex)&&(RegWriteAddr_wb==RsAddr_ex);
    assign ForwardA[1]=RegWrite_mem&&(RegWriteAddr_mem!=0)&&(RegWriteAddr_mem==RsAddr_ex);

    assign ForwardB[0]=RegWrite_wb&&(RegWriteAddr_wb!=0)&&(RegWriteAddr_mem!=RtAddr_ex)&&(RegWriteAddr_wb==RtAddr_ex);
    assign ForwardB[1]=RegWrite_mem&&(RegWriteAddr_mem!=0)&&(RegWriteAddr_mem==RtAddr_ex);


//MUX for A

    wire [31:0] A,B;
    assign A=(ForwardA[1]==0)?((ForwardA[0]==0)?(RsData_ex):(RegWriteData_wb)):(ALUResult_mem);

//MUX for B

    assign B=(ForwardB[1]==0)?((ForwardB[0]==0)?(RtData_ex):(RegWriteData_wb)):(ALUResult_mem);
```

（4）MEM 级：存储器访问级。只有在执行 LW、SW 指令时才对存储器进行读写，对其他指令只起 到缓冲一个周期的作用。该级只需要存储器写操作允许信号 MemWrite。

（5）WB 级：回写级。此级把指令执行的结果回写到寄存器文件中。该级设置信号 MemToReg 和寄 存器写操作允许信号 RegWrite，其中 MemToReg 决定写入寄存器的数据来自于 MEM 级上的缓冲值或来 自于 MEM 级上的存储器

注解几个自定义端口：

1） RegWrite 决定是否对寄存器(registers)进行写操作。 RegWrite 有效时，将数据写入指定的寄存器中

2） RegDst 决定目标寄存器是 rt 还是 rd。RegDst=0 时，rt 为目标寄存器。RegDst=1 时，rd 为目标寄存器

3） MemWrite 决定是否对数据存储器进行写操作。 MemWrite 有效时，将数据写入数据存储器指定的位置

4） MemRead 决定是否对数据存储器进行读操作。 MemRead 有效时，读取数据存储器指定位置的数据。 需要对读存储器的指令只有 LW，所以：MemRead_id= LW

5） MemtoReg 决定写入寄存器(registers)的数据来自 ALU 还是数据存储器。MemtoReg=0 时，数据来自 ALU。 MemtoReg=1 时，数据来自数据存储器

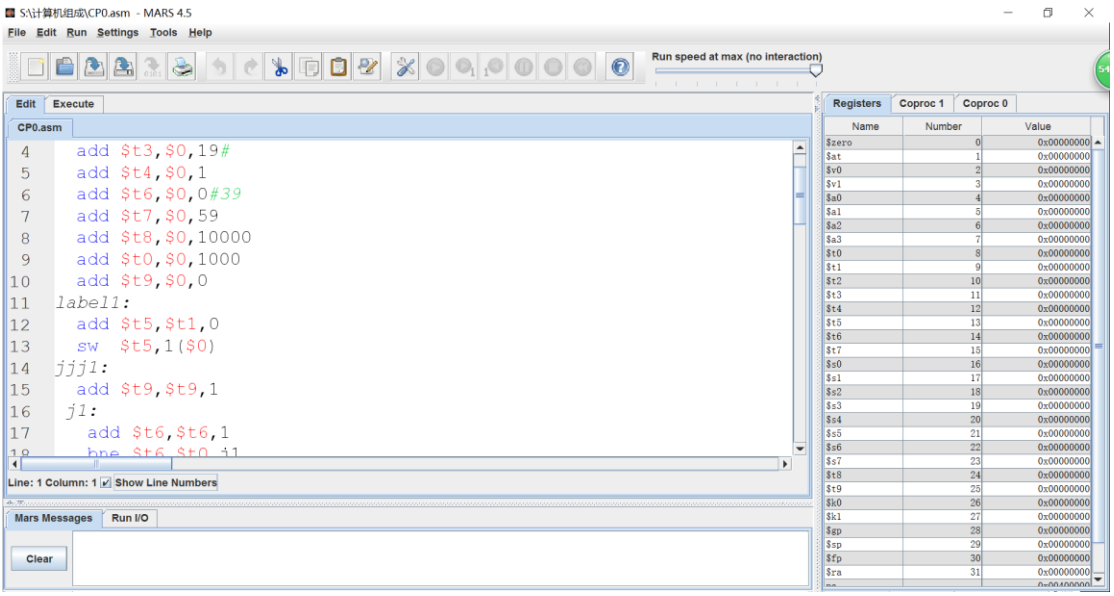
6） 其余部分直接根据名字可以识别。

四、 实验仿真过程

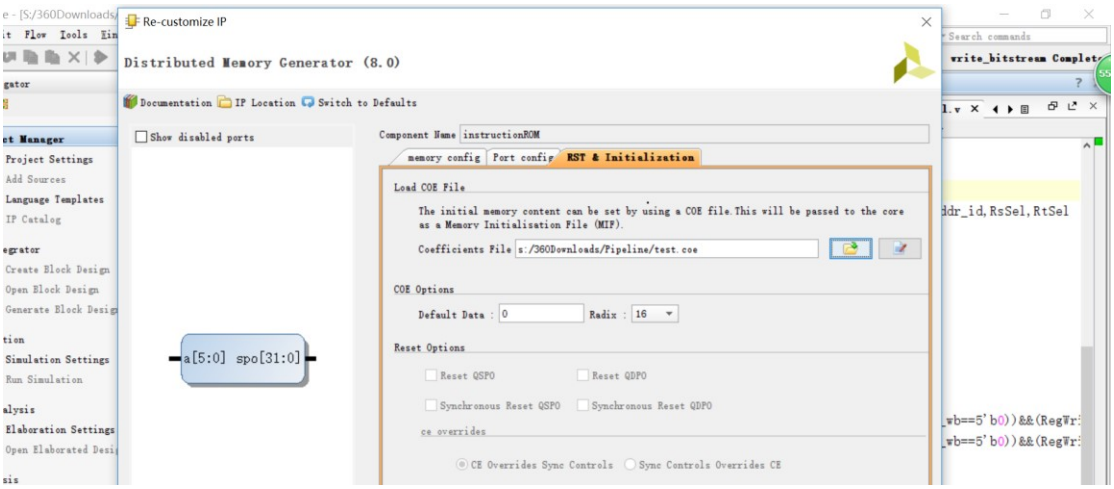
1、 静态流水线的仿真过程

4.1.1 前仿真

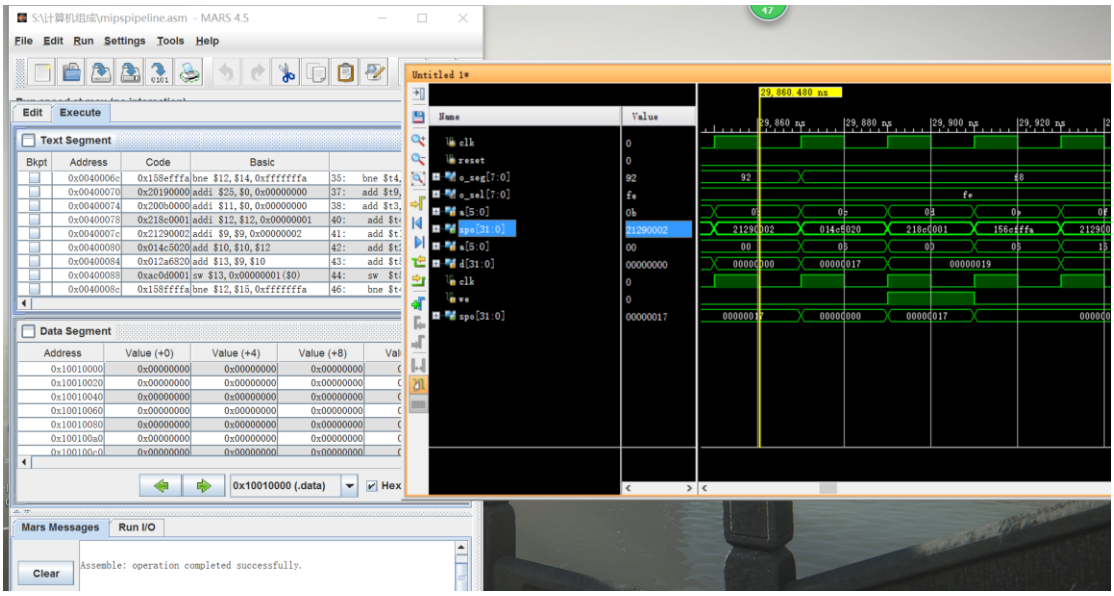
首先将测试程序写成 Mips 指令，用 mars 即可。



将得到的指令机器码，改写成 coe 文件，装入指令存储器的 ip 核中



编写 testbanch 文件，得到波形图如下：



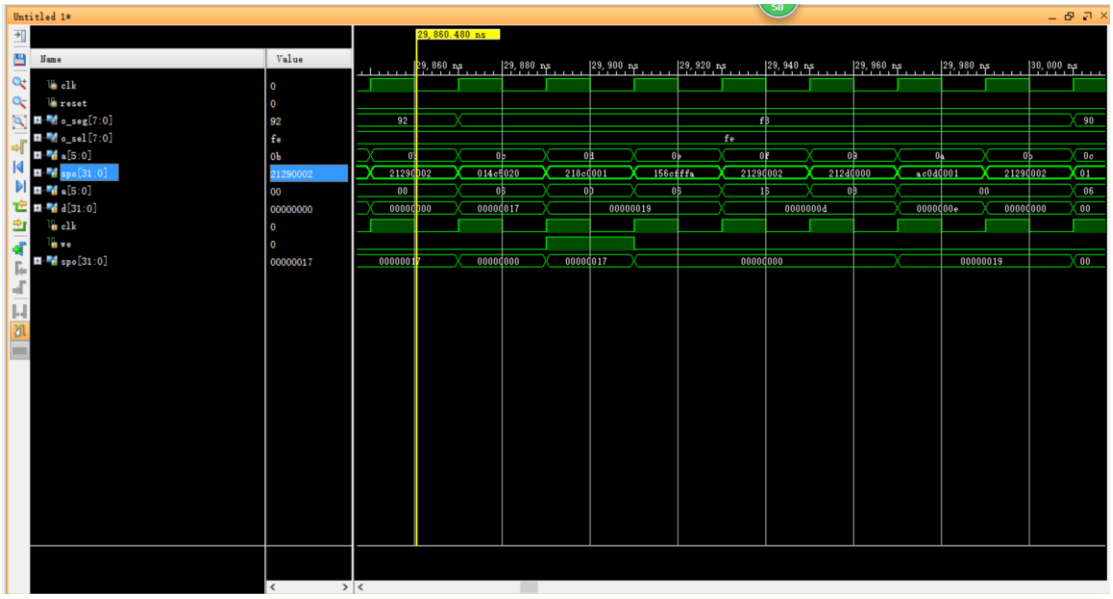
经检查指令与端口输出，无误，进行下一步后仿真以及下板。

4.1.2 后仿真

后仿真也称为时序仿真或者布局布线后仿真，是指电路已经映射到特定的工艺环境以后，

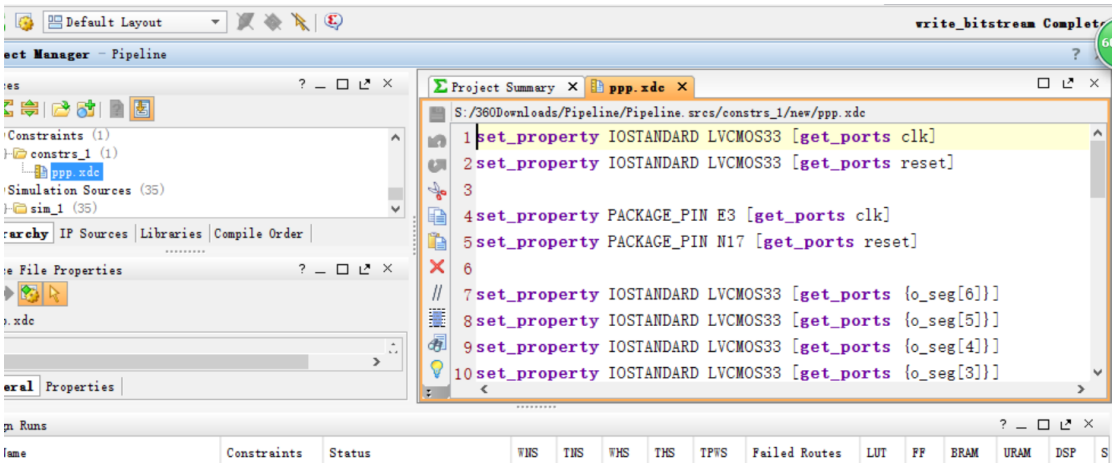
综合考虑电路的路径延迟与门延迟的影响，验证电路能否在一定时序条件下满足设计构想的 过程，是否存在时序违规。其输入文件为从布局布线结果中抽象出来的门级网表、 Testbench 和扩展名为 SDO 或 SDF 的标准时延文件。SDO 或 SDF 的标准时延文件不仅包 含门延迟，还包括实际布线延迟，能较好地反映芯片的实际工作情况。一般来说后仿真是必 选的，检查设计时序与实际的 FPGA 运行情况是否一致，确保设计的可靠性和稳定性。选 定了器件分配引脚后在做后仿真。

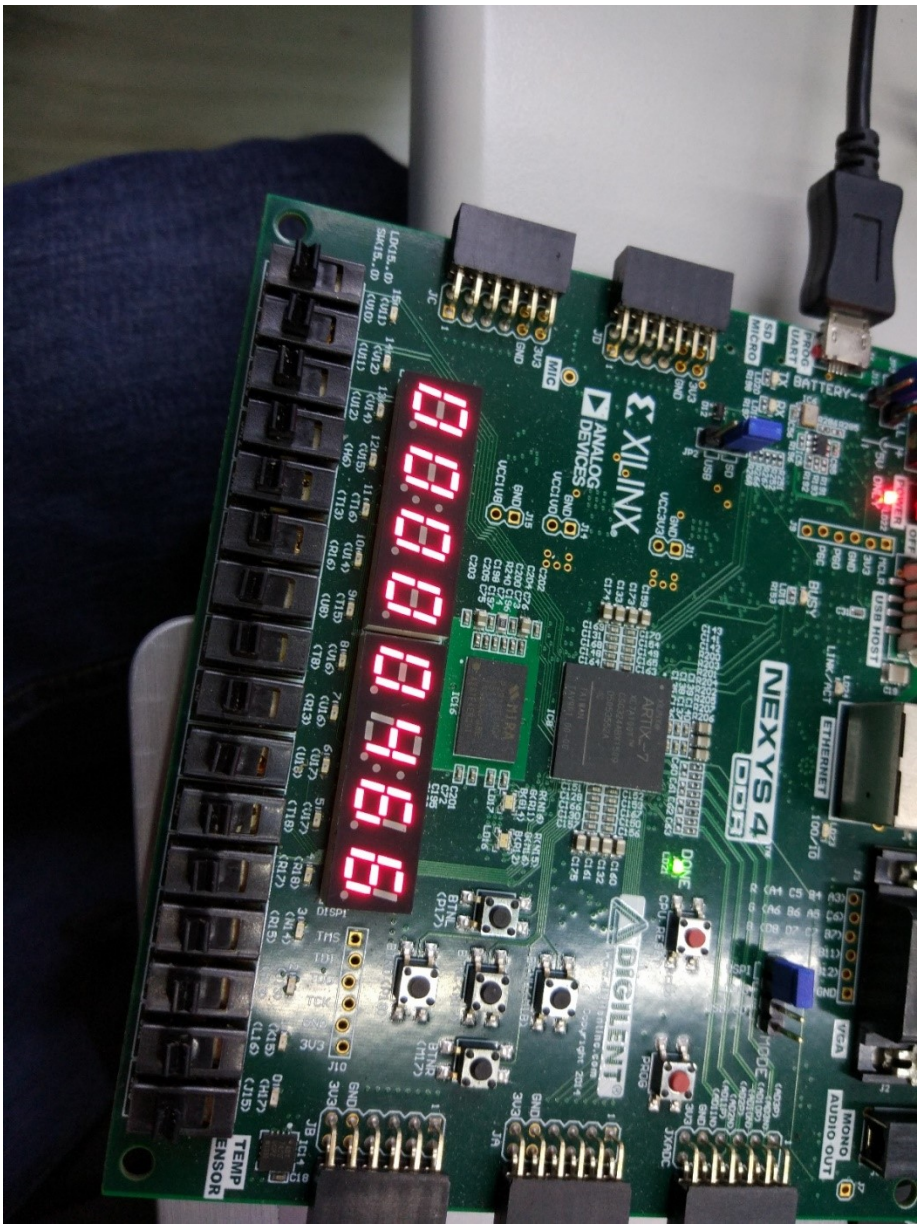
在 asm 文件中加入延迟，写入 coe 文件，后仿真如下图。



4.1.3 下板综合

下板： 加入 seg7*16 文件，设置好管教约束文件

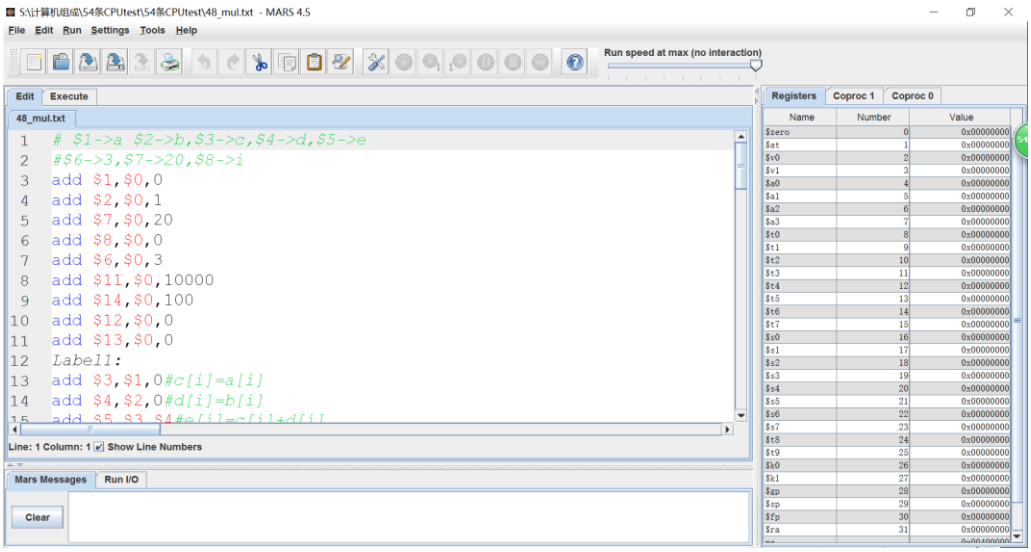


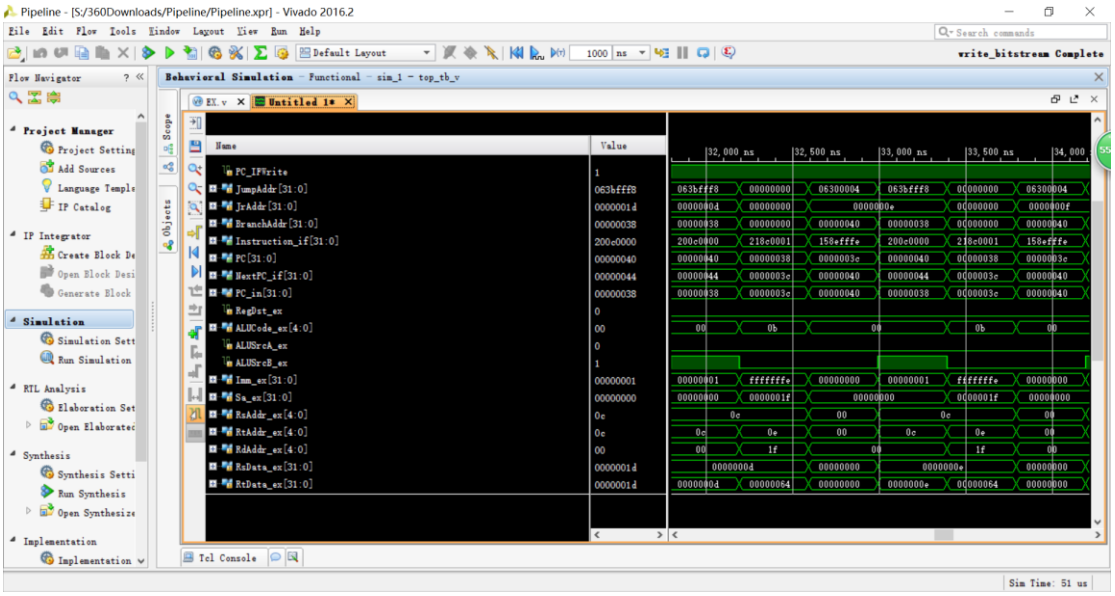


2、 动态流水线的仿真过程

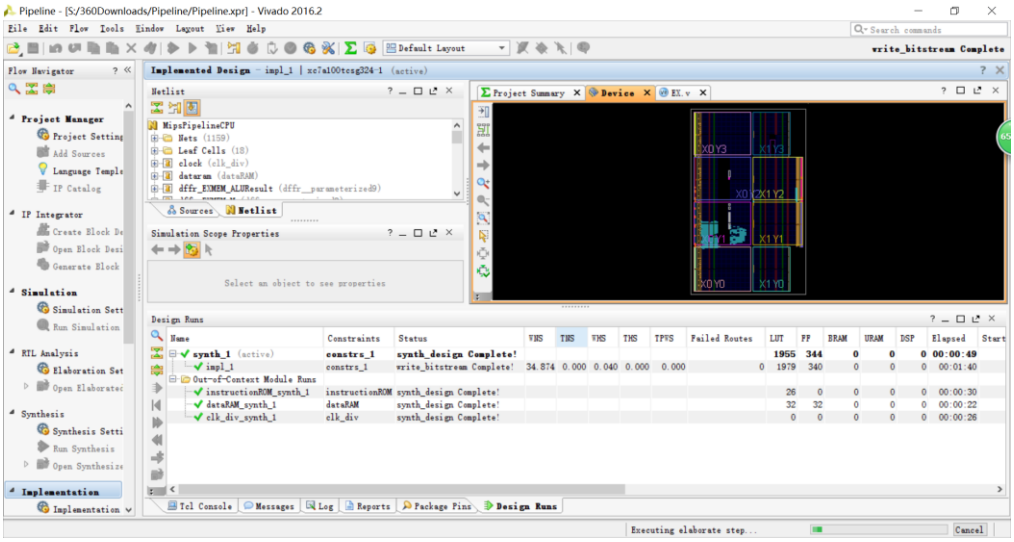
4.2.1 前仿真

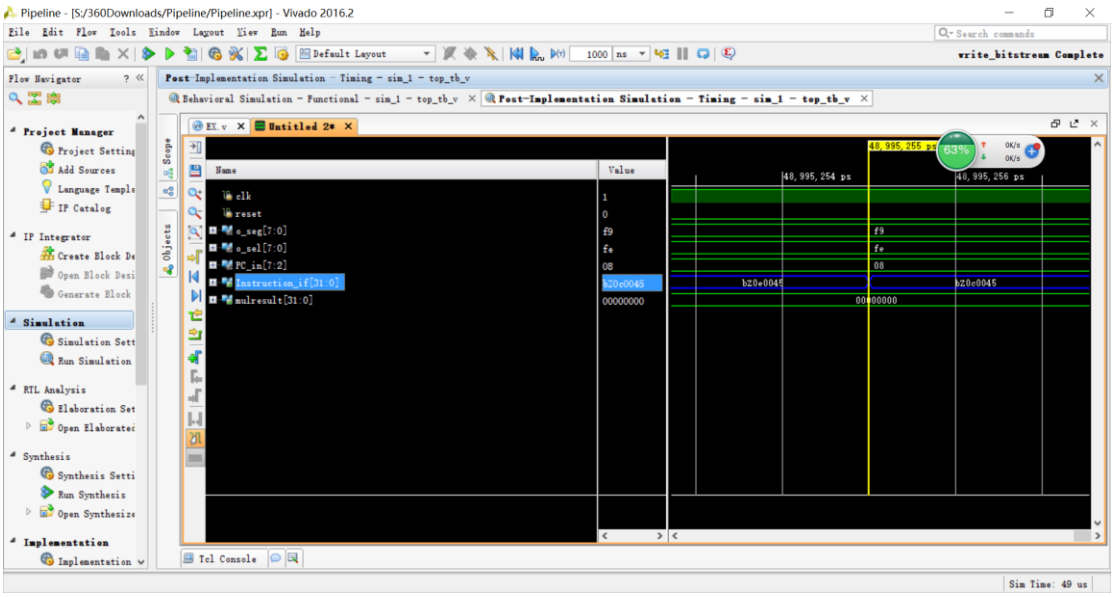
同静态流水线一样


S:\计算机组成\54条CPUtest\54条CPUtest\48_mul.txt - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
48_mul.txt
1 # $1->a $2->b,$3->c,$4->d,$5->e
2 #$6->3,$7->20,$8->i
3 add $1,$0,0
4 add $2,$0,1
5 add $7,$0,20
6 add $8,$0,0
7 add $6,$0,3
8 add $11,$0,10000
9 add $14,$0,100
10 add $12,$0,0
11 add $13,$0,0
12 Label1:
13 add $3,$1,0#c[i]=a[i]
14 add $4,$2,0#d[i]=b[i]
Line: 1 Column: 1 Show Line Numbers
Registers Coproc 1 Coproc 0
Name Number Value
Mars Messages Run I/O
Clear

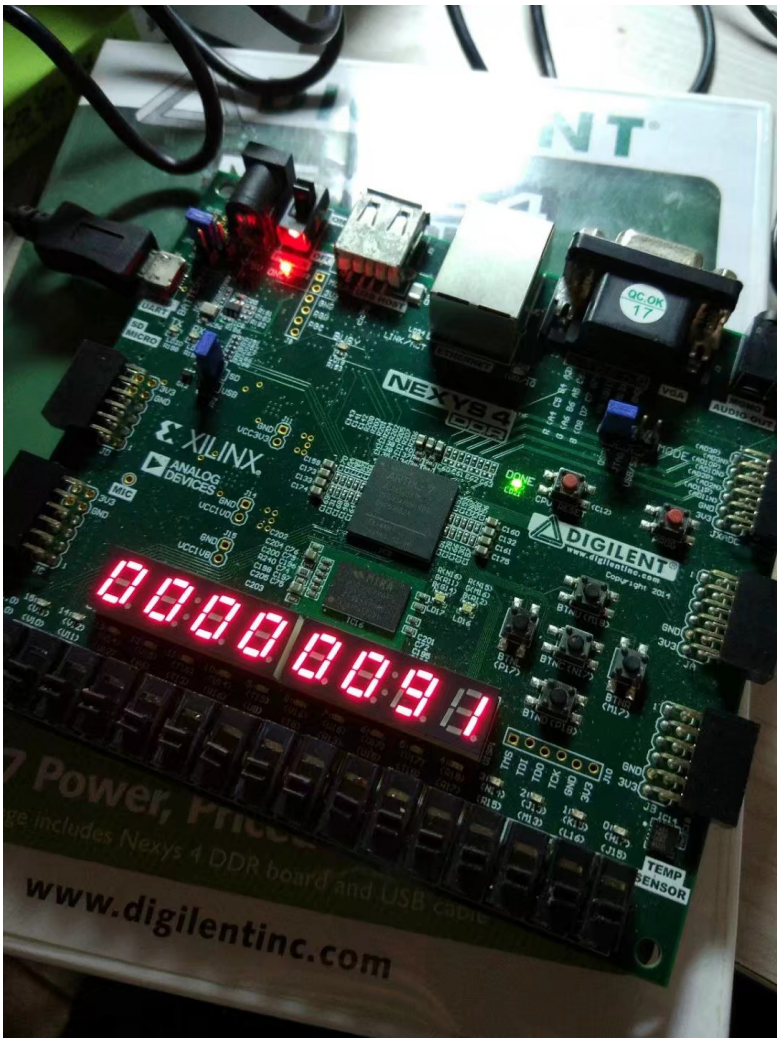

Pipeline - [S:/360Downloads/Pipeline/Pipeline.xpr] - Vivado 2016.2
File Edit Flow Tools Window Layout View Run Help
write_bitstream Complete
Behavioral Simulation - Functional - sim_1 - top_tb_v
Untitled 1
Flow Navigator
Project Manager
IP Integrator
Simulation
RTL Analysis
Synthesis
Implementation
Tcl Console
Sim Time: 51 us

4.2.2 后仿真


Pipeline - [S:/360Downloads/Pipeline/Pipeline.xpr] - Vivado 2016.2
File Edit Flow Tools Window Layout View Help
write_bitstream Complete
Implemented Design - impl_1
Flow Navigator
Project Manager
IP Integrator
Simulation
RTL Analysis
Synthesis
Implementation
Design Runs
Tcl Console Messages Log Reports Package Pins Design Runs
Executing elaborate step...
Cancel

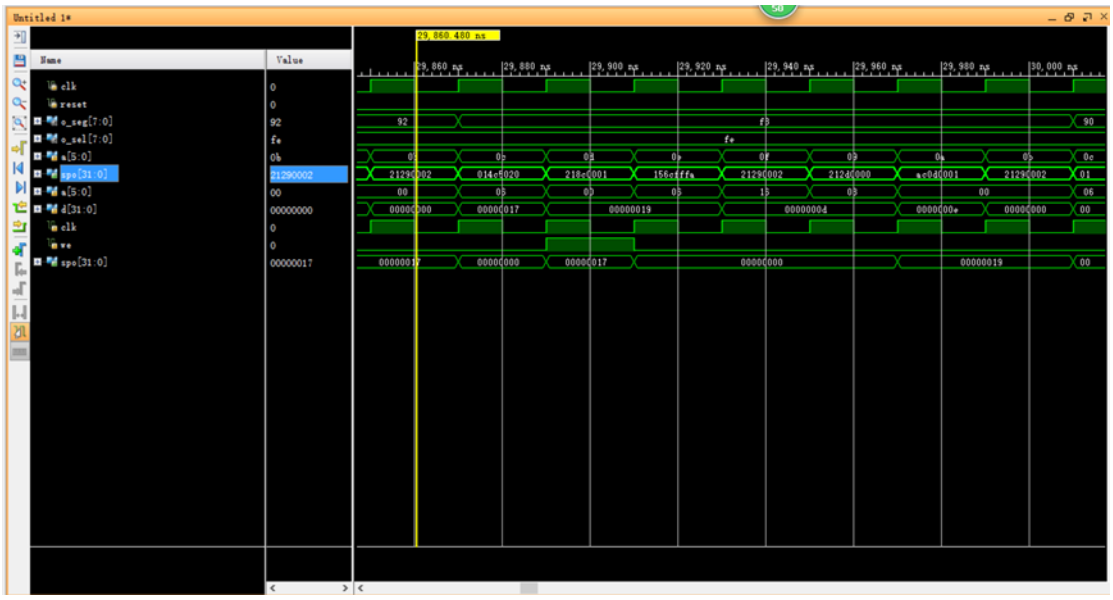


4.2.3 下板综合

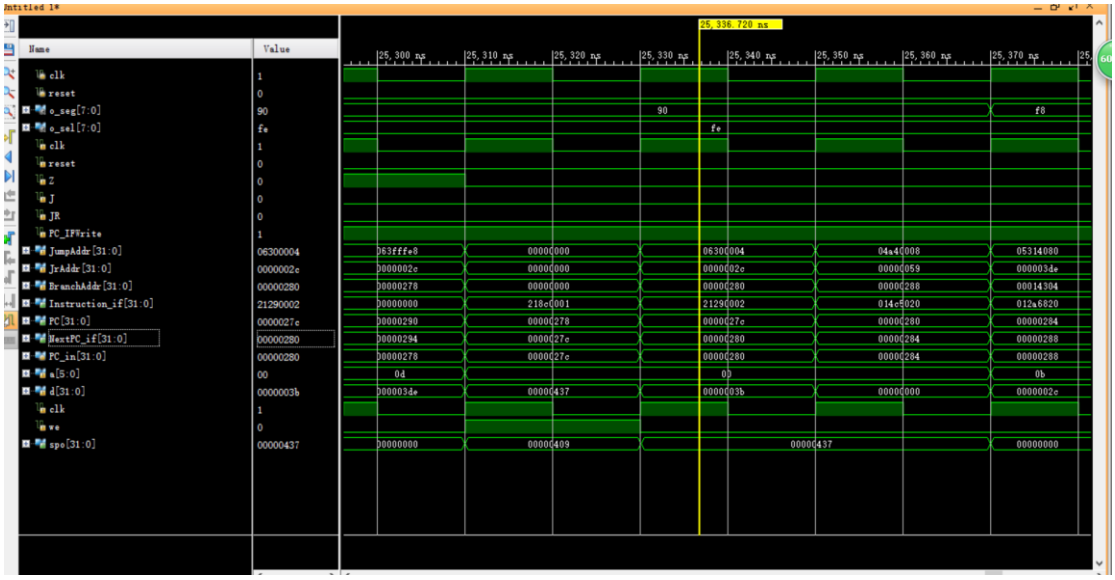


五、 实验仿真的波形图及某时刻寄存器值的物理意义

1、 静态流水线的波形图及某时刻寄存器值的物理意义

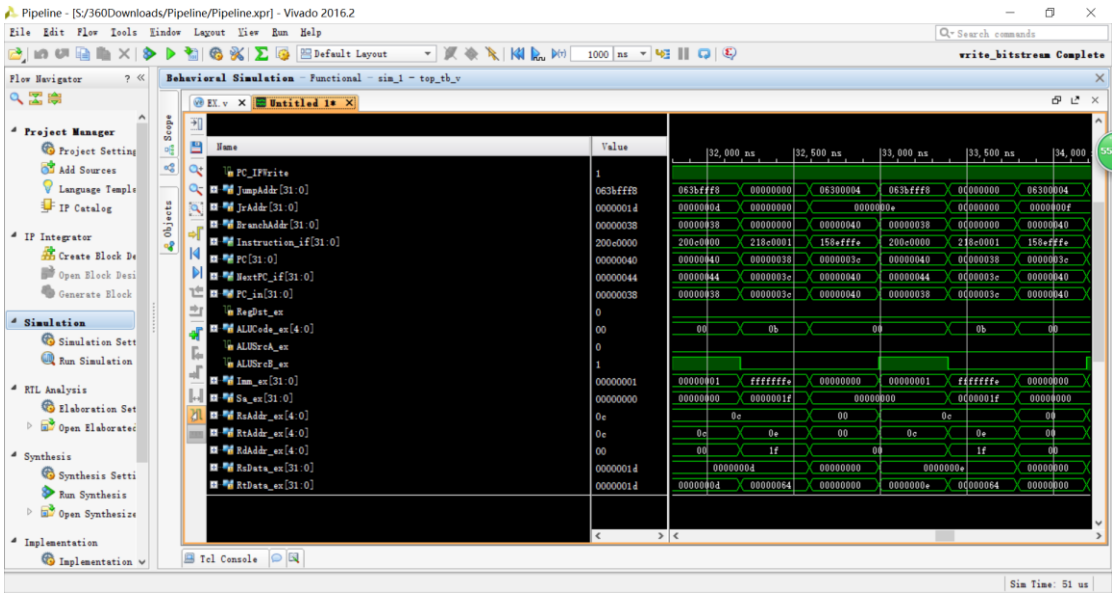


在本截图中，s-seg 和 s-sel 是有关数码管的数值，第一个 spo 是指令存储器的输出



在此图中，除了以上外还有 JumpAddr(跳转地址),JrAddr(jr 指令地址)，BranchAddr(分支地址)，Instruction_if（IF 阶段的指令值），PC,nextpc_if(if 阶段的下一个 pc)

2、 动态流水线的波形图及某时刻寄存器值的物理意义



Pc_ifwrite 是 pc 寄存器的使能信号

JumpAddr 是跳转地址，JrAddr 是 Jr 指令的地址，BranchAddr 是分支地址

Instruction_if 是译码阶段的指令等等

六、 实验验算数学模型及算法程序

Int a[m],b[m],c[m],d[m]
a[0]=0;b[0]=1;
a[i]=a[i-1]+I;b[i]=b[i-1]+3*i
c[i]=a[i],d[i]=b[i](0<=i<=19)
c[i]=a[i]+b[i],d[i]=c[i]*a[i](20<=i<=39)
c[i]=a[i]*b[i];d[i]=c[i]*b[i](40<=i<=59)

程序：

```
# $1->a $2->b,$3->c,$4->d,$5->e
#$6->3,$7->20,$8->i
add $1,$0,0
add $2,$0,1
add $7,$0,20
add $8,$0,0
add $6,$0,3
add $11,$0,10000
add $14,$0,100
add $12,$0,0
add $13,$0,0
```

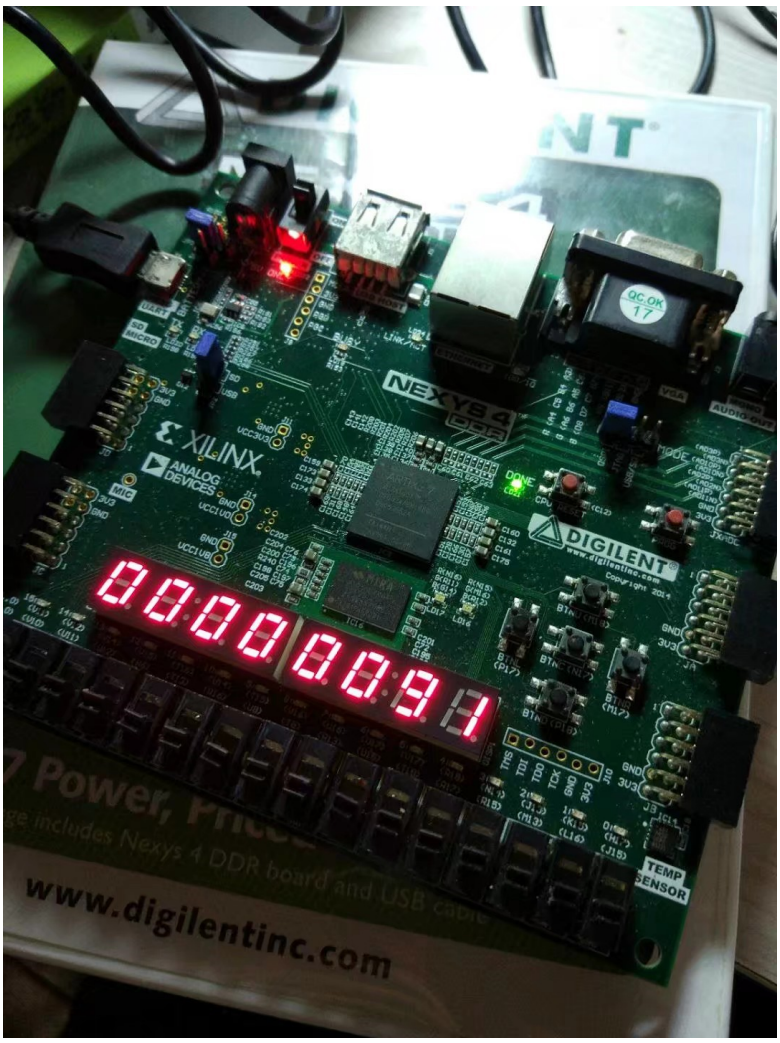
```
Label1:
add $3,$1,0#c[i]=a[i]
add $4,$2,0#d[i]=b[i]
add $5,$3,$4#e[i]=c[i]+d[i]
sw $5,0
#loop
loop1:
add $13,$13,1
loop11:
add $12,$12,1
bne $12,$14,loop11
add $12,$0,0
bne $13,$11,loop1
add $13,$0,0
add $8,$8,1#i++
add $1,$1,$8#a[i+1]=a[i]+i+1
mul $9,$8,$6#b[i+1]=b[i]+3*(i+1)
add $2,$2,$9
bne $7,$8,Label1

add $7,$0,40
add $13,$0,0
add $12,$0,0
Label2:
add $3,$1,$2#c[i]=a[i]+b[i]
mul $4,$3,$1#d[i]=c[i]*a[i]
add $5,$3,$4#e[i]=c[i]+d[i]
sw $5,0
#loop
loop2:
add $13,$13,1
loop21:
add $12,$12,1
bne $12,$14,loop21
add $12,$0,0
bne $13,$11,loop2
add $13,$0,0
add $8,$8,1#i++
add $1,$1,$8#a[i+1]=a[i]+i+1
mul $9,$8,$6#b[i+1]=b[i]+3*(i+1)
add $2,$2,$9
bne $8,$7,Label2

add $7,$0,60
```

```
        add $13,$0,0
        add $12,$0,0
        Label3:
        mul $3,$1,$2
        mul $4,$3,$2
        add $5,$3,$4
        sw $5,0
        #loop
        loop3:
        add $13,$13,1
        loop31:
        add $12,$12,1
        bne $12,$14,loop31
        add $12,$0,0
        bne $13,$11,loop3
        add $13,$0,0
        add $8,$8,1#i++
        add $1,$1,$8#a[i+1]=a[i]+i+1
        mul $9,$8,$6#b[i+1]=b[i]+3*(i+1)
        add $2,$2,$9
bne $7,$8,Label3
```

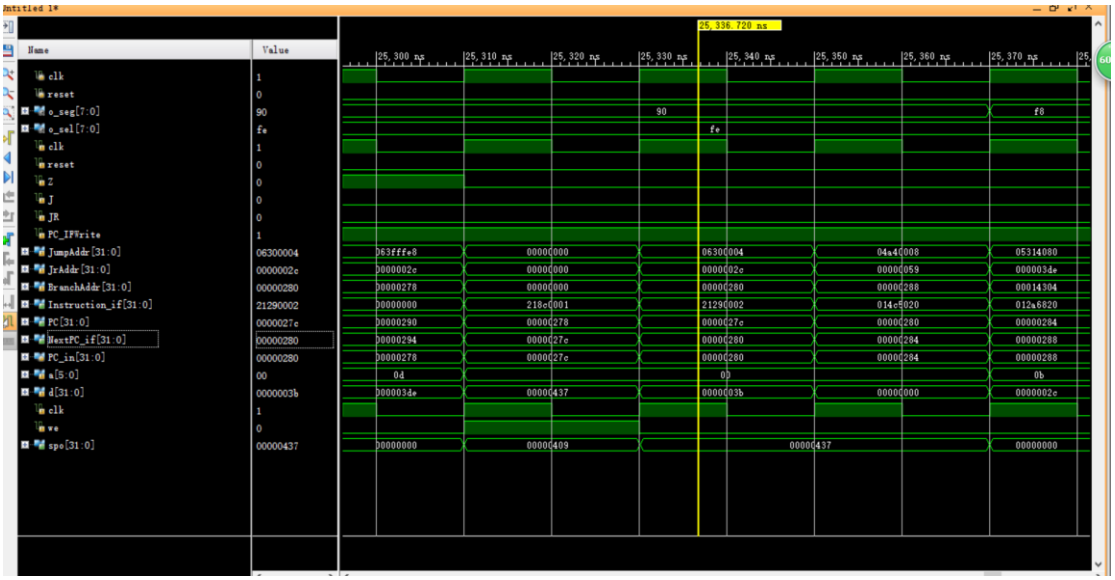
七、 序下板测试过程与实现



(中间得数截图)

八、 流水线的性能指标定性分析

1、 静态流水线的性能指标定性分析



由于在 Mips 指令中我加入了死循环（便于显示最后得数）,所以截取 200 个周期，得出 70 个结果，所以

吞吐率是=70/(200*Δt)

加速比=324Δt/200 Δt=1.62

效率=324/1000=0.324

冲突时采用气泡暂停法和延迟分支

```
    output timeout;
//  reg x;
    assign stall=MemRead_ex&&((RegWriteAddr_ex==RsAddr_id)||(RegWriteAddr_e
    assign PC_IFWrite=~stall;
    assign timeout=(stall==1)?3:0;
```

2、 动态流水线的性能指标定性分析

同理由于在 Mips 指令中我加入了死循环（便于显示最后得数）,所以截取 200 个周期，得出 121 个结果

吞吐率=121/(200 Δt)

加速比=524 Δt/200 Δt=2.62

效率=524/1000=0.524

冲突采用数据转发（数据定向），引入流水线阻塞，即气泡（bubble），延迟分支方法。

```
//forwarding
    wire[1:0] ForwardA,ForwardB;

    assign ForwardA[0]=RegWrite_wb&&(RegWriteAddr_wb!=0)&&(RegWriteAddr_mem!=RsAddr_ex)&&(RegWriteAddr_wb==RsAddr_ex);
    assign ForwardA[1]=RegWrite_mem&&(RegWriteAddr_mem!=0)&&(RegWriteAddr_mem==RsAddr_ex);

    assign ForwardB[0]=RegWrite_wb&&(RegWriteAddr_wb!=0)&&(RegWriteAddr_mem!=RtAddr_ex)&&(RegWriteAddr_wb==RtAddr_ex);
    assign ForwardB[1]=RegWrite_mem&&(RegWriteAddr_mem!=0)&&(RegWriteAddr_mem==RtAddr_ex);


//MUX for A

    wire [31:0] A,B;
    assign A=(ForwardA[1]==0)?((ForwardA[0]==0)?(RsData_ex):(RegWriteData_wb)):(ALUResult_mem);

//MUX for B

    assign B=(ForwardB[1]==0)?((ForwardB[0]==0)?(RtData_ex):(RegWriteData_wb)):(ALUResult_mem);
```

3、两者性能差异分析

处理多种运算时可以看出，动态流水线的性能远远优于静态流水线，静态流水线必须等一种运算的流水线排空才能进行另一种运算，动态流水线却可以允许某些段在实现某种运算时，另一些段却在实现另外一种运算，因而对于本次数学模型中有加有乘，动态流水线会更加优越。
但是如果是单一运算，静态流水线则更为常见，因为动态流水线控制过于复杂。

九、 总结与体会

设计思想来自课本以及《计算机原理与设计》（并且这本书有代码），很多时候都是参考李亚民老师的代码，看起来很清楚，设计的时候，各种端口复杂交错，我以前设计的端口和指导书上对应不起来，不得不推翻以前 54 条的顶层模块，重新设计，底层模块由于基本一样，所以不需要大改。

通过这次设计，我理解了三种冲突，以及对应的解决方案，对流水线 MIPS CPU 的特点在本次实验中得到了充分体现，虽然最终完成的 CPU 的功能还比较简单，但对于理论的学习加深印象的同时，也让我学到了很多理论课学不到的东西，更加细节性的东西通过自己 的动手有了较深的印象，但是也体会到了体系结构的艰辛，底层硬件设计思想与软件思想的区别，动手能力进一步提高了。课外花费时间还是很大的，几周的午睡没了，晚上也睡得很晚，一个小小的 wire 和 reg 的区别带来的 bug 都能让我查很久（vivado 调 bug 的能力要有 vs 一半就好了）。

十、 附件（所有程序）

见附件。