《数字电路》课程实验

实验2：CPU设计

----实验过程与结果分析报告

选课编号：XX 姓名（邓皓宇）选课号（XX）

选课编号：XX 姓名（熊仔羿）选课号（XX）

信息与通信工程学院

2022年11月

## 实验目的

1.掌握简化CPU的结构组成、模块划分和工作原理。

2.掌握精简指令集的概念，并设计CPU的指令集。

3.用Verilog语言设计编写并调试CPU内部各模块电路，并在此基础上设计完整CPU。

4.掌握激励文件的编写方法，学会验证各个模块功能时序的正确性。

5.掌握外围接口电路的设计方法，并学会利用FPGA开发板的按键、LED等外设构造简单的控制系统。

6.编写汇编程序，并依据设计的指令集编译为对应的机器码，控制CPU实现特定功能。

## 二、实验内容

任务1：基础模块（ALU、寄存器阵列、寻址单元）的建模及仿真

任务2：数据通路的设计

任务3：控制通路的设计

任务4：CPU的整合及验证

## 三、实验设备

软件： PC 机操作系统 WinXP 或 Win7、 Quartus II 13.1、 ModelSim-Altera 10.1d

硬件： DE1-SOC 实验板，计算机一台

仪器：示波器，逻辑分析仪

## 四、实验原理

任务1

**目标：** **基础模块（ALU、寄存器阵列、寻址单元）的建模及仿真**

**任务计划：**

课前预习任务：算术逻辑单元模块（ArithmeticUnit.v）设计与仿真。 （检查）  
课上实验任务：寄存器阵列模块（RegisterFile.v）设计与仿真。 （检查）

**实验原理：**CPU的数据通路是用来控制CPU内部组件之间的数据传输和信号传输，以实现特定的功能。数据通路的主要组件包括ALU（算数逻辑单元）、控制器和存储器，它们之间通过总线或紧凑的总线实现数据传输，并根据控制器发出的指令控制ALU进行计算。数据通路有实现加法、减法、乘法、除法、移位、比较等各种操作，从而实现计算机系统的指令及其对应的运算。

任务2

**目标：**

**任务计划：**

课前预习任务： 寻址单元（AddressLogic.v， ProgramCounter.v） 设计与仿真。  
（检查）  
课上实验任务：数据通路（DataPath.v）整合。 （检查）

**实验原理：**CPU的寻址单元是一个硬件部件，用于识别存储单元的地址，也称为地址翻译单元（ATU）或寻址翻译器（ATL）。它根据指令中的源地址和目的地址，对对应的存储位置进行翻译。寻址单元是计算机内部运算的基础，是CPU实现内部指令执行、控制指令传递和数据传递的关键部分。

任务3

**目标：** 实现一个控制通路，能够完成指令的译码和执行。并且能够争取读取数据并存回寄存器。完成control.v，并且实现相应的testbench。

**任务计划：** 按照PPT和指导书上的指示，自行完成mil,mih,mvr三个指令的verilog代码。

课前预习任务：预习实验指导书控制通路

课上实验任务：（检查）

**实验原理：**状态机特别适合描述发生有先后顺序或者有逻辑规律的事情，故一般控制通路 采用状态机来描述。控制通路是CPU中的控制通路负责读取程序中的指令，并根据指令的类型和内容控制其他部件的运作方式。控制通路还负责将执行的指令的结果写入内存中，以便供程序的其他部分使用。它的工作原理是读取程序中的指令，根据指令的类型和内容选择执行的操作，并控制各个部件的运作方式。例如，如果指令要求将一个数字加上另一个数字，控制通路就会控制单元（ALU）去完成这个操作。控制通路还会控制程序计数器（PC）去读取下一条指令，以便处理程序中的其他指令。控制单元向外部存储器发送读数据请求。外部存储器根据地址输入的数据 定位读数据的位置，取出指令数据。

任务4

**目标：** 完成整个CPU的整合，并且编写相应的机器码，实现流水灯功能。

**任务计划：** 完成整个CPU的编写，分析综合，编写汇编代码，并且翻译为机器码，写入RAM并执行。外部编写PLL分频电路，设置外部复位功能。

**实验原理：**建立cpu\_top模块，链接CPU相关的连线，并且向外暴露时钟，复位，输入，输出端口。CPU复位后从RAM读入指令开始执行。启东时，初始化寄存器00 01，并且开始左移计数，当计数到一定数量后清零，以实现LED流水灯的功能。

课前预习任务：预习实验指导书CPU的整合

课上实验任务：（检查）整合CPU并进行仿真分析，下板验证

**实验原理：**建立cpu\_top模块，链接CPU相关的连线，并且向外暴露时钟，复位，输入，输出端口。CPU复位后从RAM读入指令开始执行。启东时，初始化寄存器00 01，并且开始左移计数，当计数到一定数量后清零，以实现LED流水灯的功能。

## 五、实验步骤

**任务1的步骤如下：**

1. 第一步

使用QuartusⅡ13.1打开工程文件，并且打开ArithmeticUnit.v,AddressLogic.v， ProgramCounter.v和RegisterFile.v文件，书写空缺处的Verilog代码

1. 第二步

写完代码更换顶层文件，之后对整个工程进行编译，并且生成Testbench(.vt)文件

1. 第三步

编写Testbench，并且配置好仿真设置

1. 第四步

启动Modelsim软件，并且观察仿真现象。

**任务2的步骤如下：**

1. 第一步

使用QuartusⅡ13.1打开工程文件，并且打开DataPath.v文件，书写Verilog代码

1. 第二步

写完代码更换顶层文件，之后对整个工程进行编译，并且生成Testbench(.vt)文件

1. 第三步

编写Testbench，并且配置好仿真设置

1. 第四步

启动Modelsim软件，并且观察仿真现象。

**任务3的步骤如下：**

1. 第一步

使用QuartusⅡ13.1打开工程文件，并且打开AddressLogic.v， ProgramCounter.v文件，书写Verilog代码

1. 第二步

写完代码更换顶层文件，之后对整个工程进行编译，并且生成Testbench(.vt)文件

1. 第三步

编写Testbench，并且配置好仿真设置

1. 第四步

启动Modelsim软件，并且观察仿真现象。

**任务4的步骤如下：**

1. 第一步

使用QuartusⅡ13.1打开工程文件，并且打开AddressLogic.v， ProgramCounter.v文件，书写Verilog代码

1. 第二步

写完代码更换顶层文件，之后对整个工程进行编译，并且生成Testbench(.vt)文件

1. 第三步

编写Testbench，并且配置好仿真设置

1. 第四步

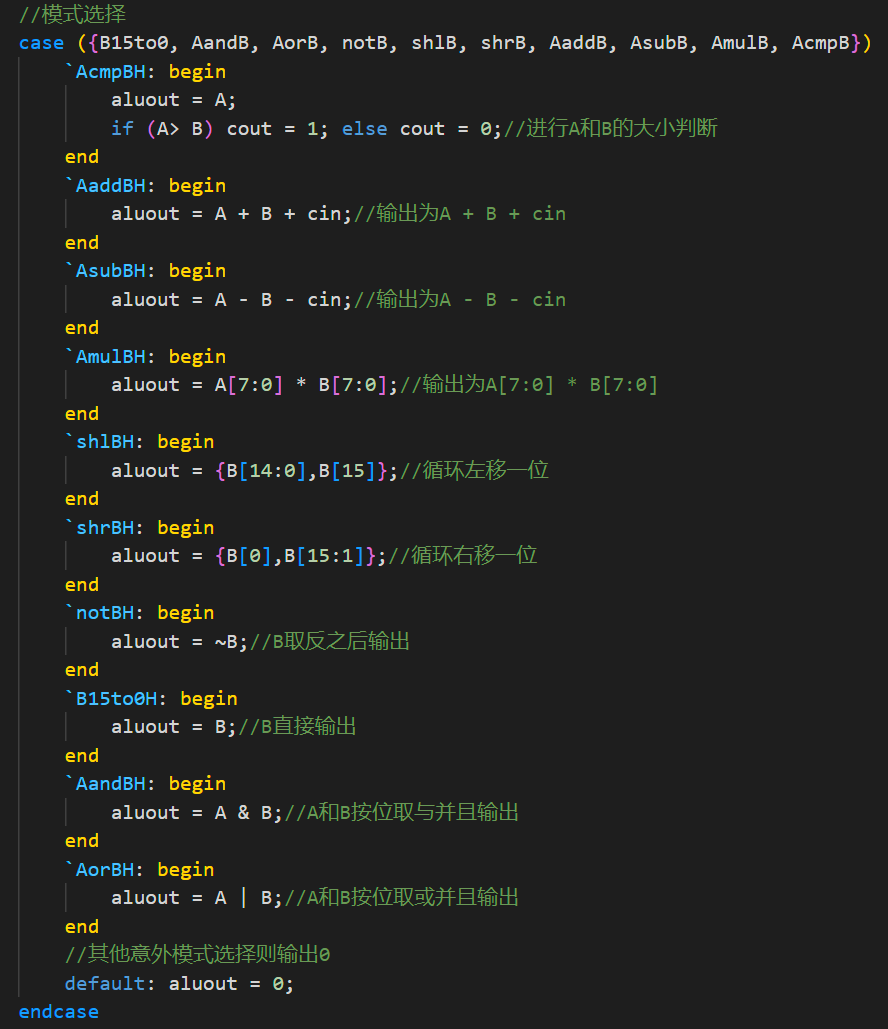
启动Modelsim软件，并且观察仿真现象。

## 六、实验结果

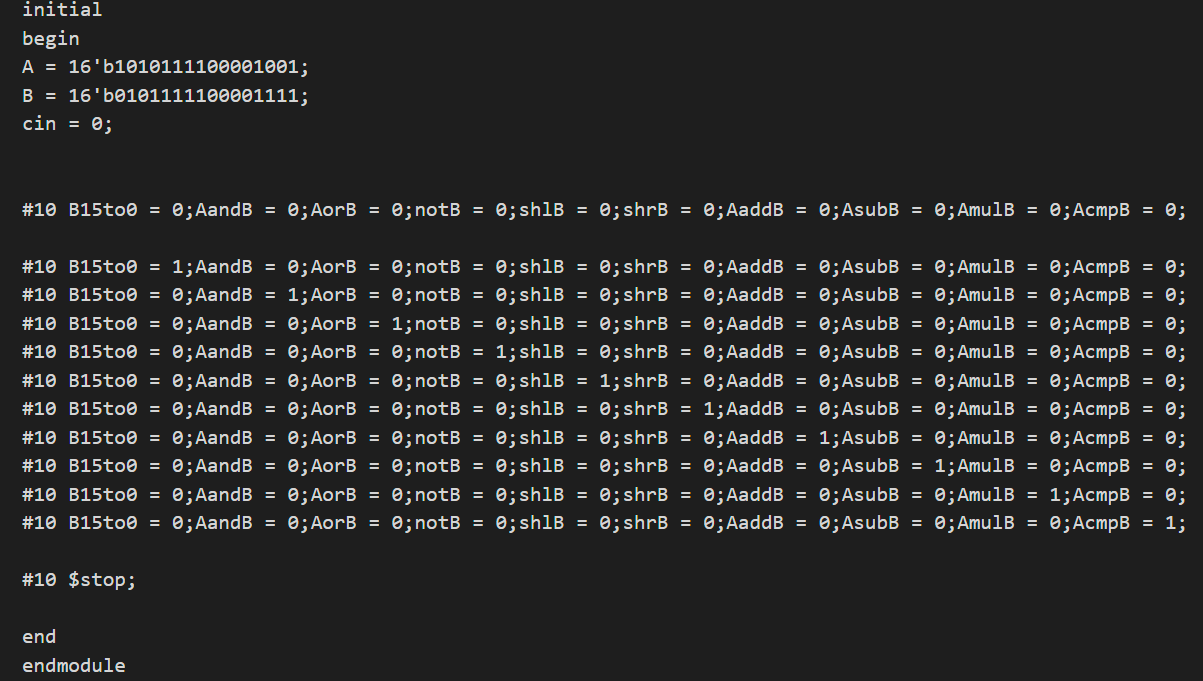
**任务1**

ALU：

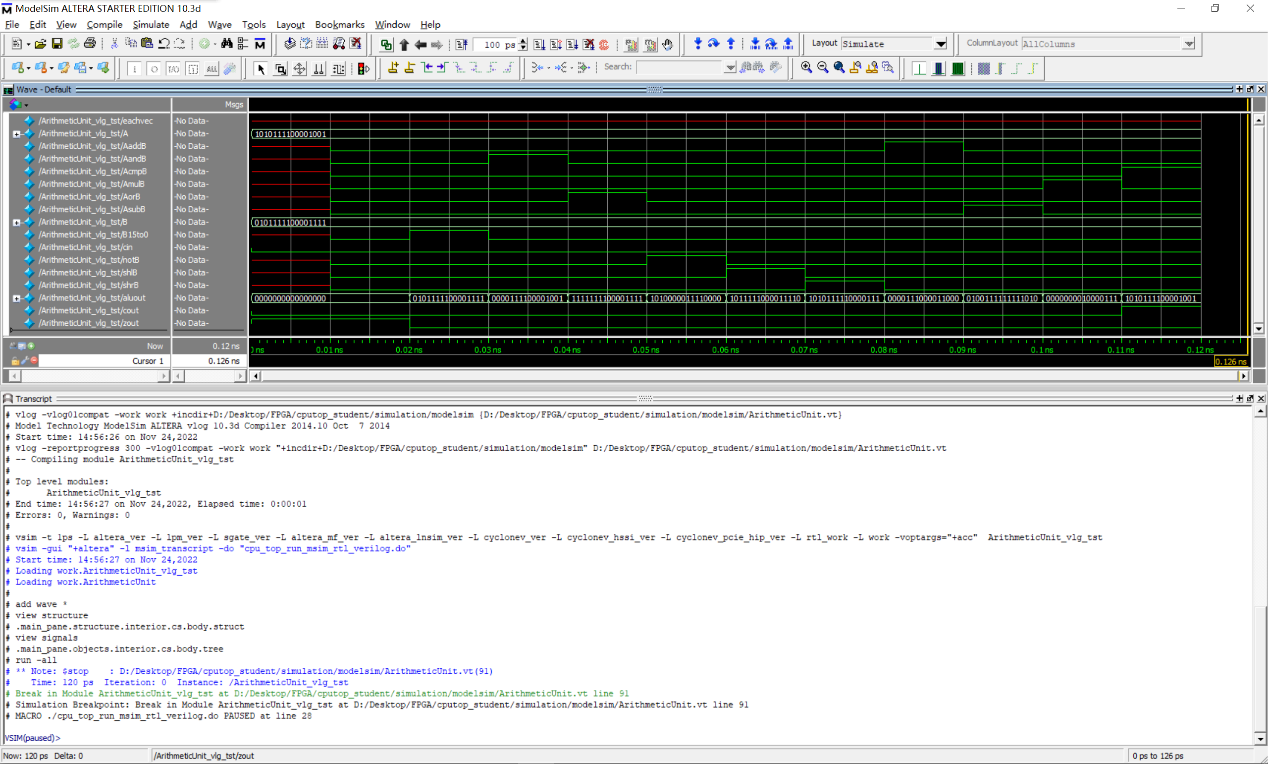
1. verilog关键代码（带注释）



1. testbench测试文件



3、仿真结果截图（带文字分析）



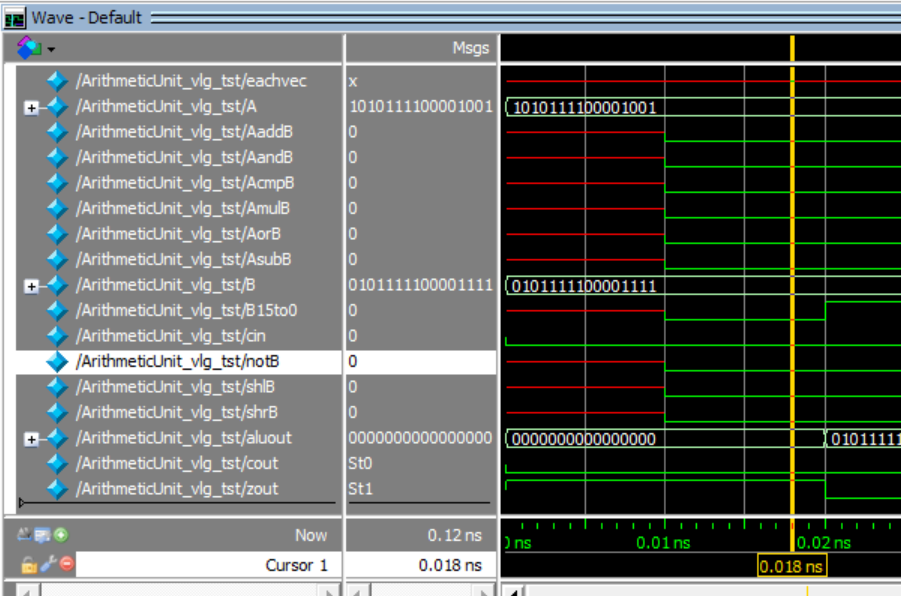
进行仿真后的波形如图所示，下面开始进行分析

1. 初始化状态



如图所示，在初始化状态时没有模式选择输入，只有对于A，B的赋值以及状态位的初始化。所以其他的线都是红色，表示未知态x。

1. 模式初始化阶段



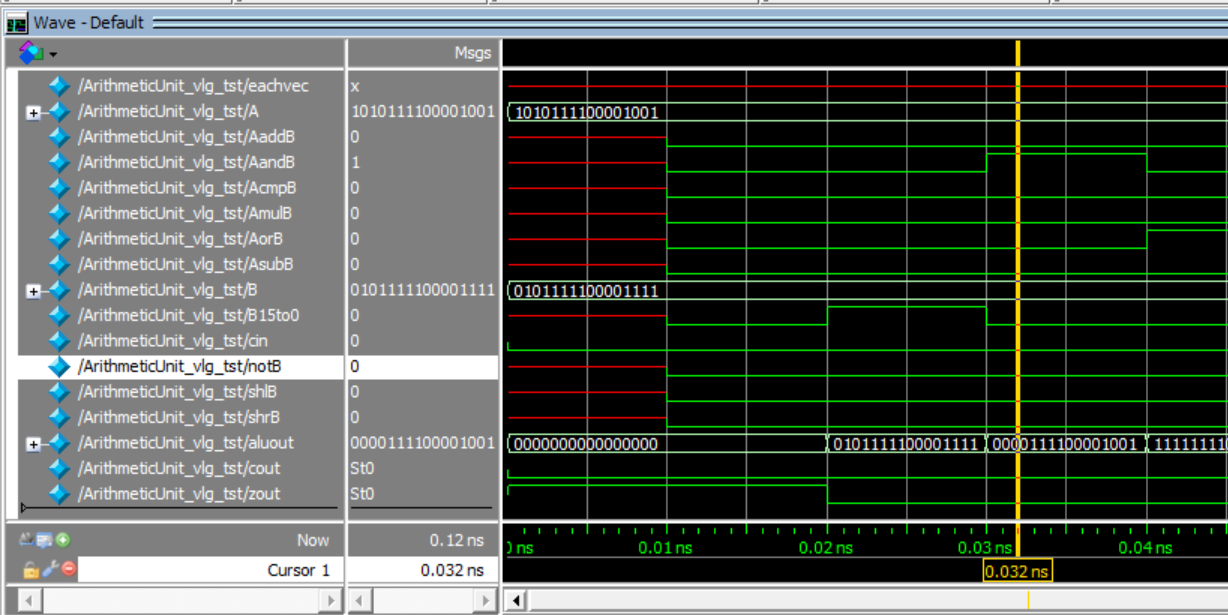
此时对于模式选择输入进行了初始化，即先全部置0，此时无输出

1. 直接输出B模式



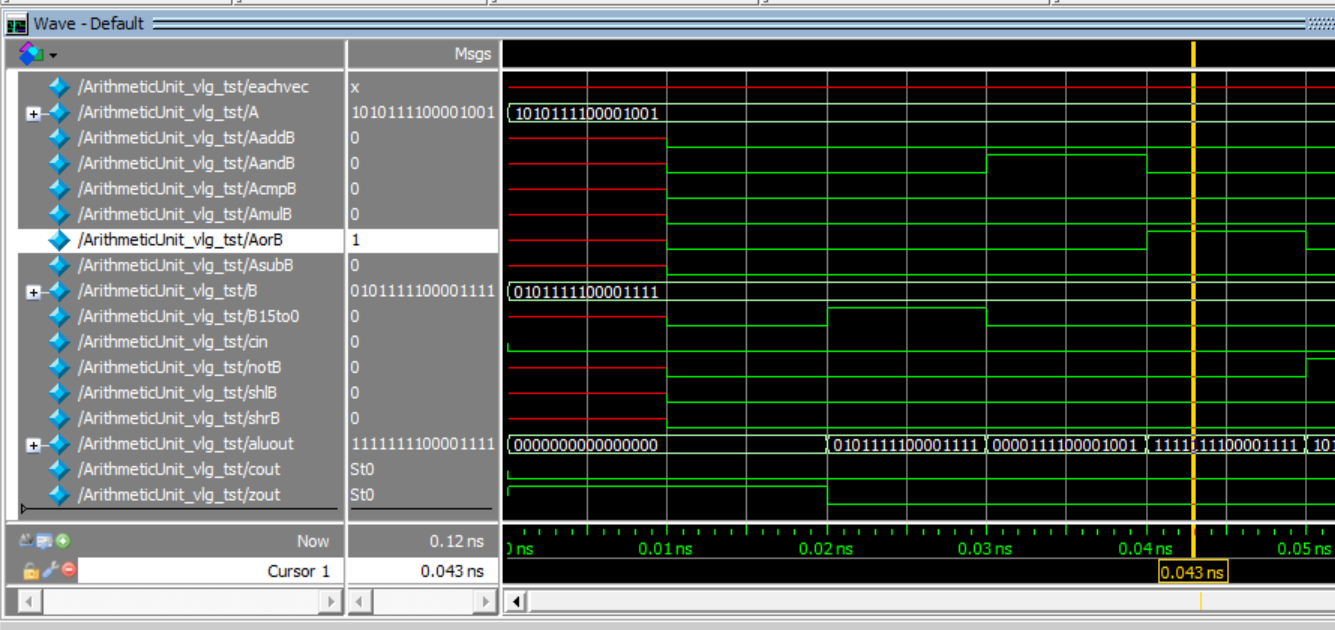
可以看到，现在输出值就是B[15:0]，其他模式为0

1. A和B按位与模式



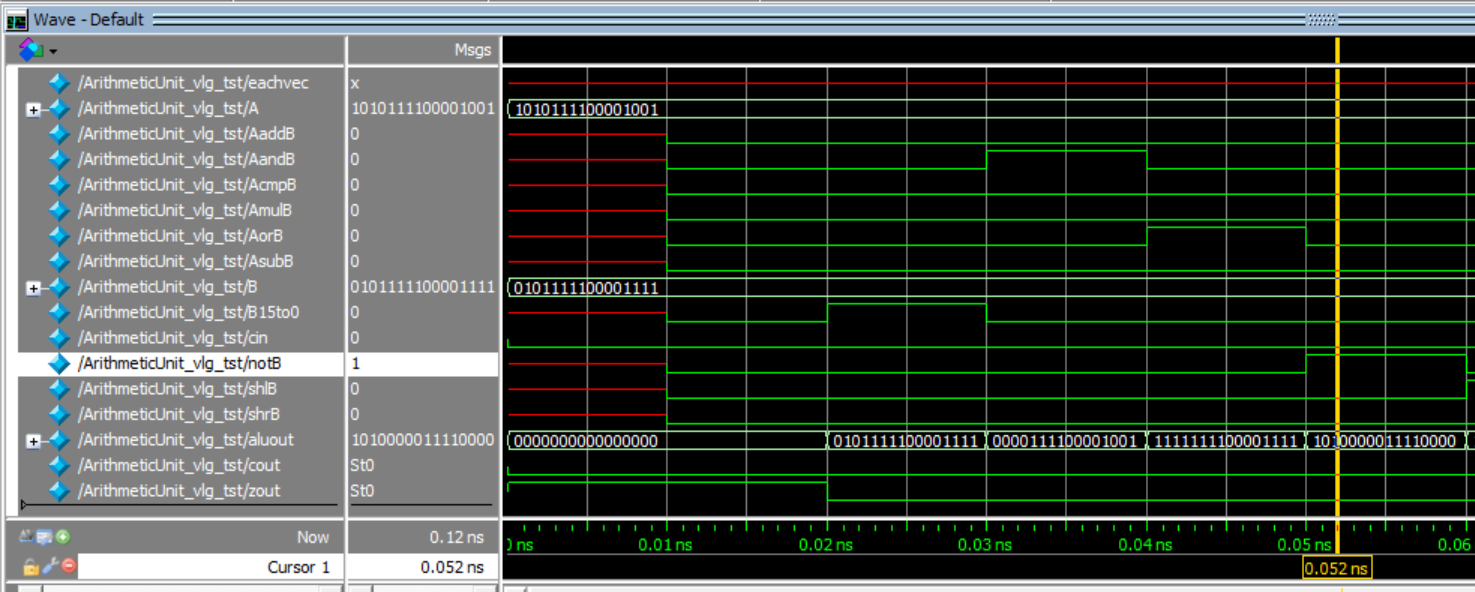
现在可以看到选取的是A与B按位与模式，输出则是A和B按位与运算的结果

1. A和B按位或模式



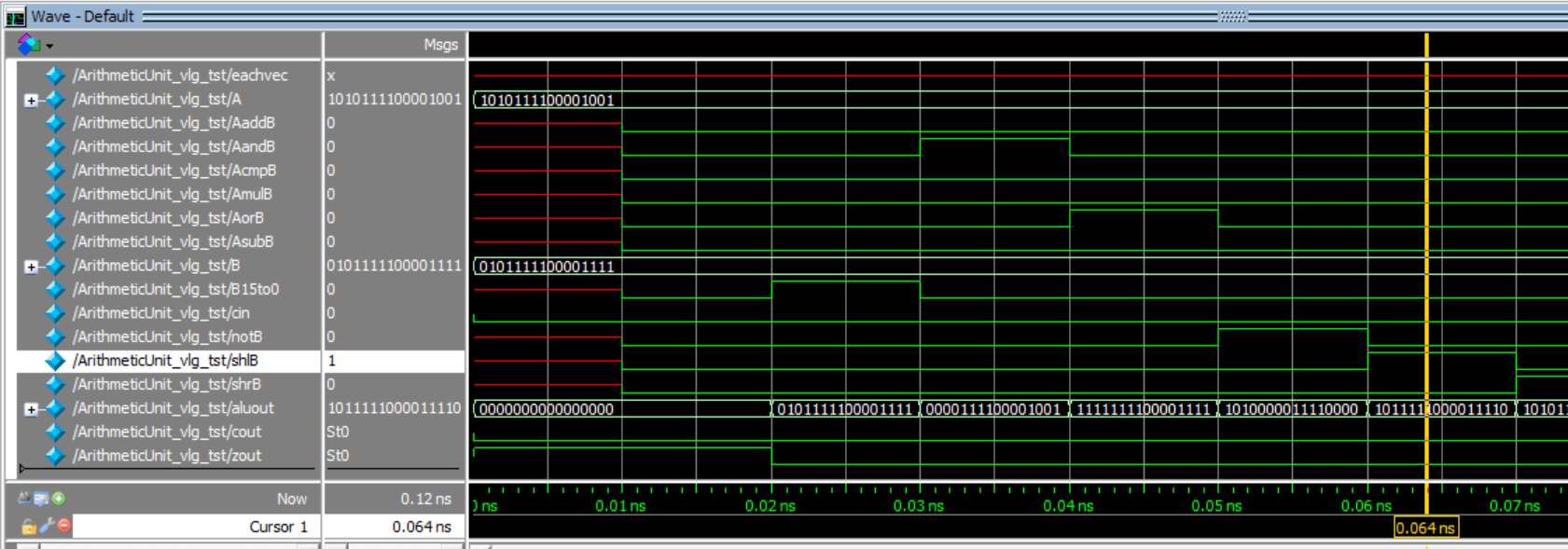
现在可以看到选取的是A与B按位或模式，输出则是A和B按位或运算的结果

1. A与B按位取反模式



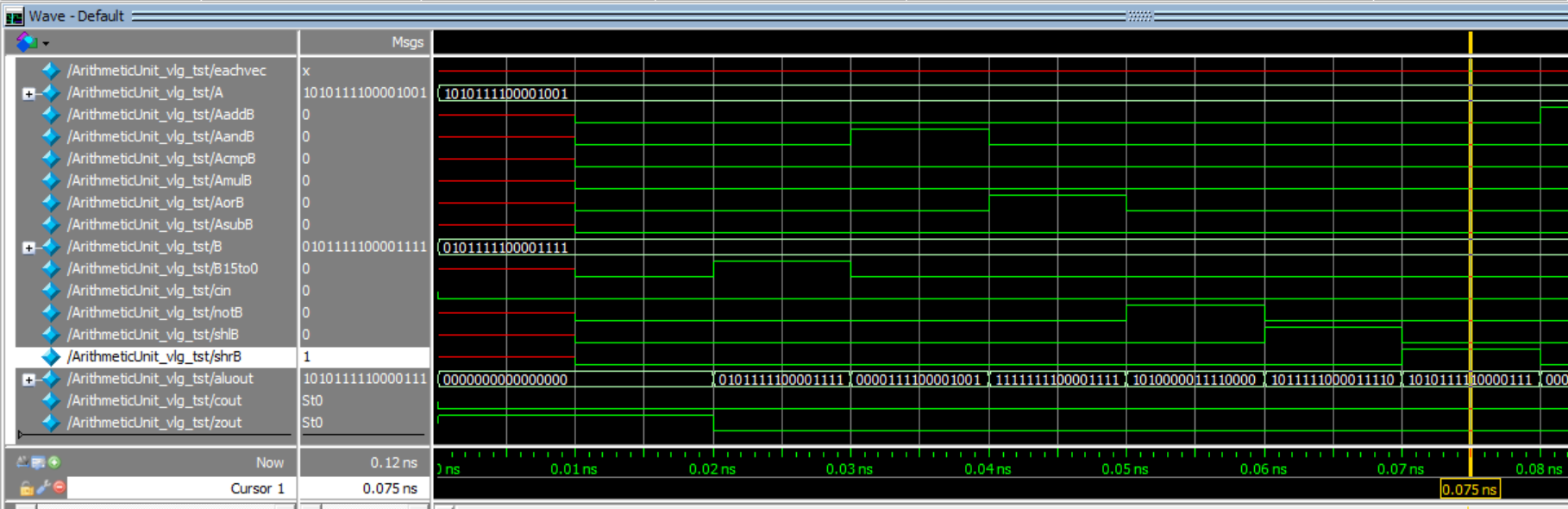
现在可以看到选取的是B按位取反模式，输出则是B按位取反运算的结果

1. 循环左移模式



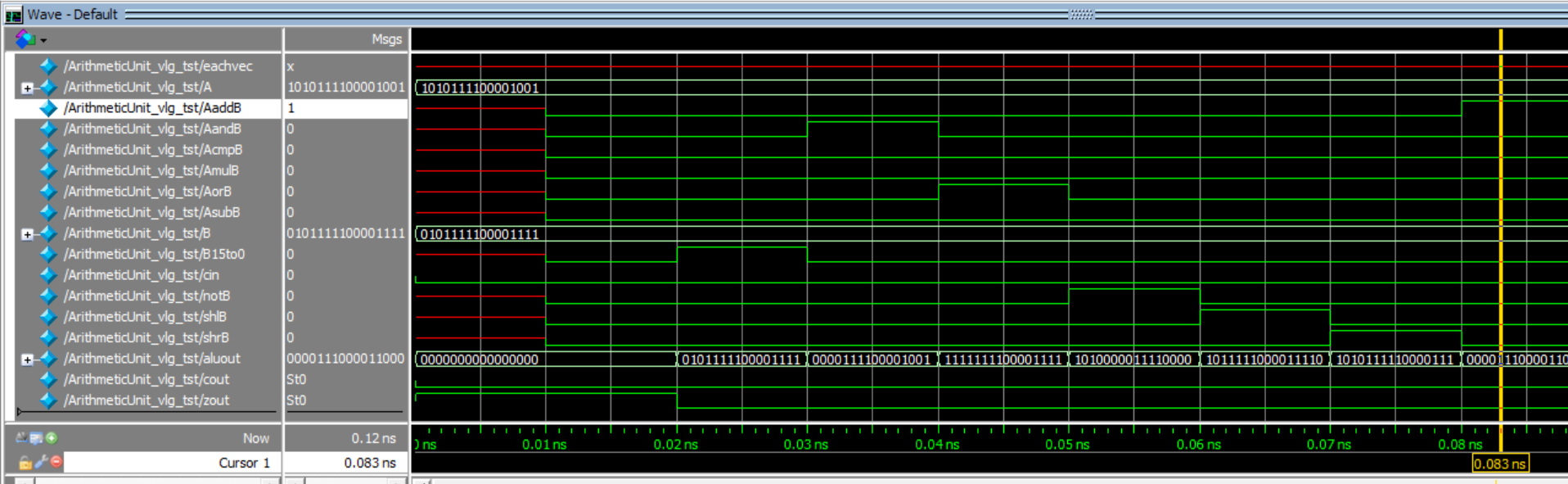
现在可以看到选取的是循环左移模式，输出则是B循环左移运算的结果

1. 循环右移模式



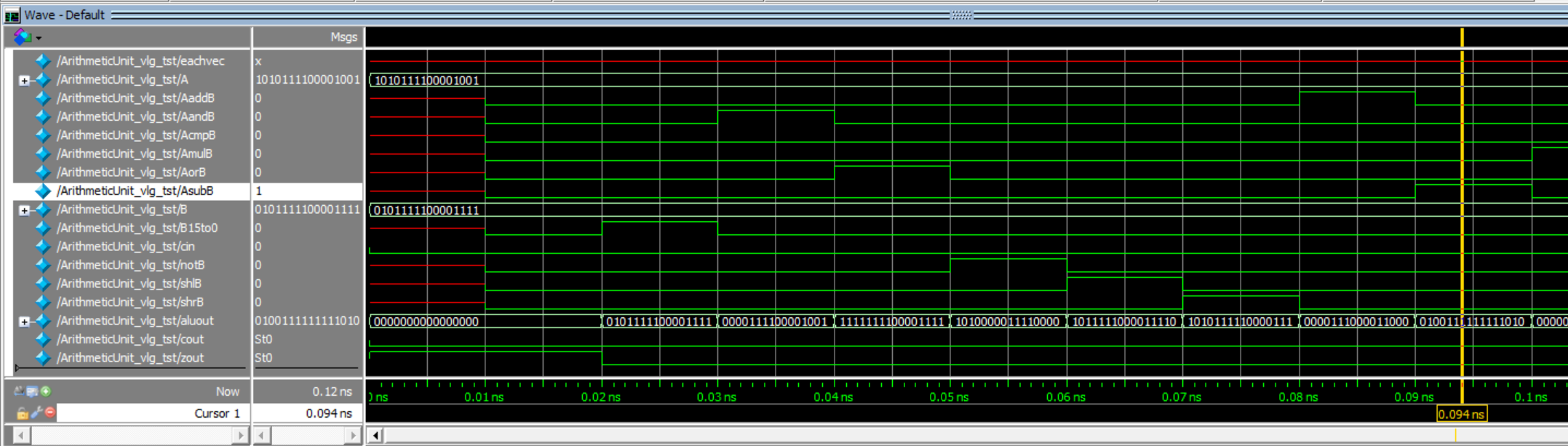
现在可以看到选取的是循环右移模式，输出则是B循环右移运算的结果

1. A+B模式



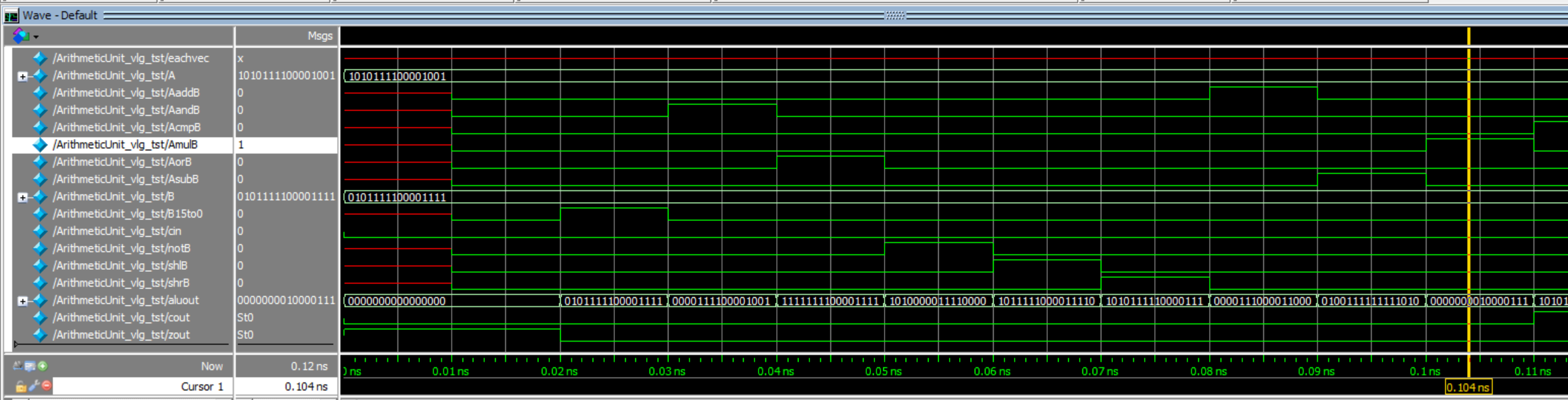
现在可以看到选取的是A+B模式，输出则是A+B运算的结果

1. A-B模式



现在可以看到选取的是A-B模式，输出则是A-B运算的结果

1. A\*B模式



现在可以看到选取的是A\*B模式，输出则是A\*B运算的结果

1. A和B比较模式



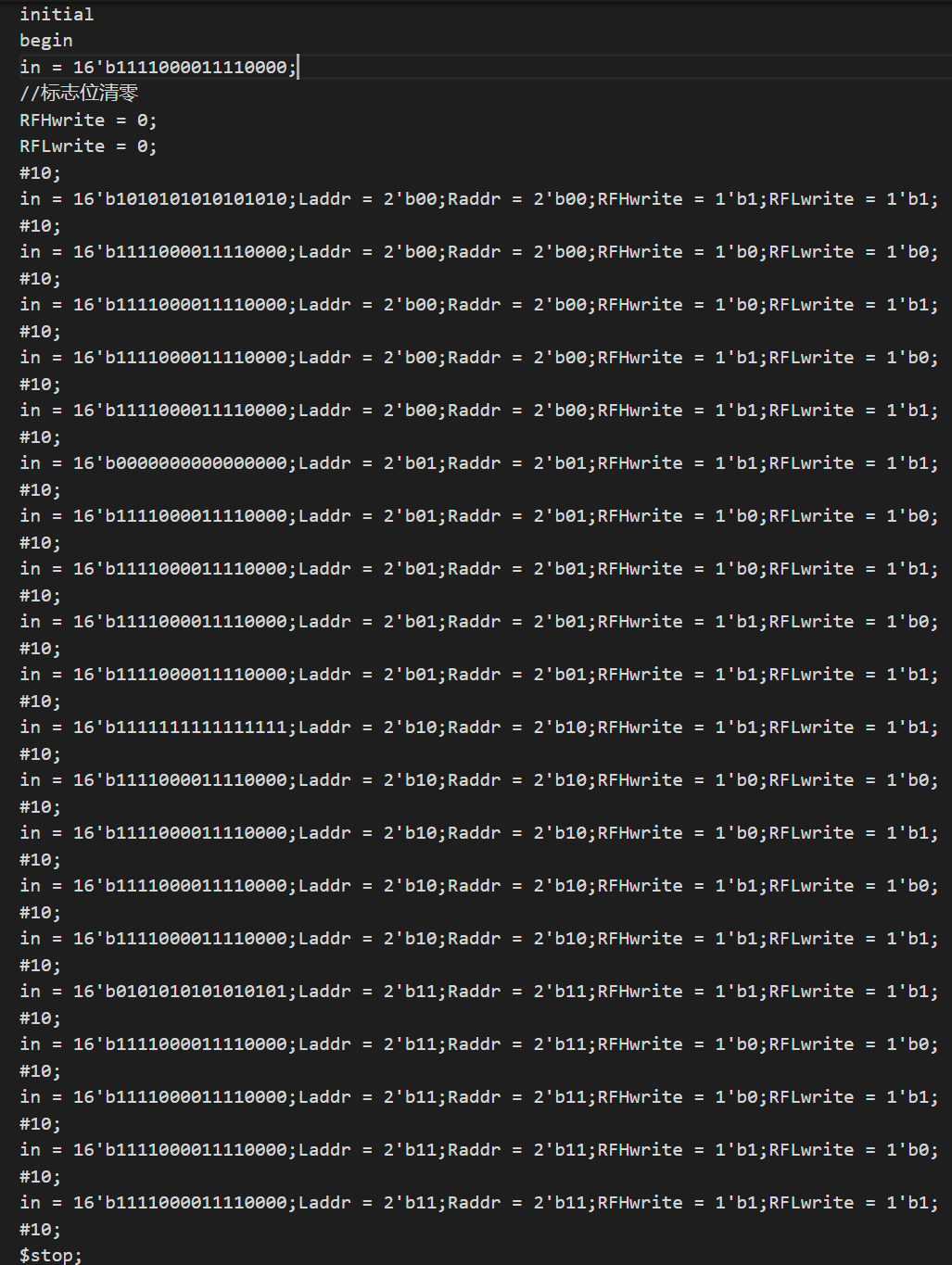
现在可以看到选取的是A和B进行比较模式，输出则是A，count现在是１，符合预期.至此，仿真结束。

寄存器阵列：

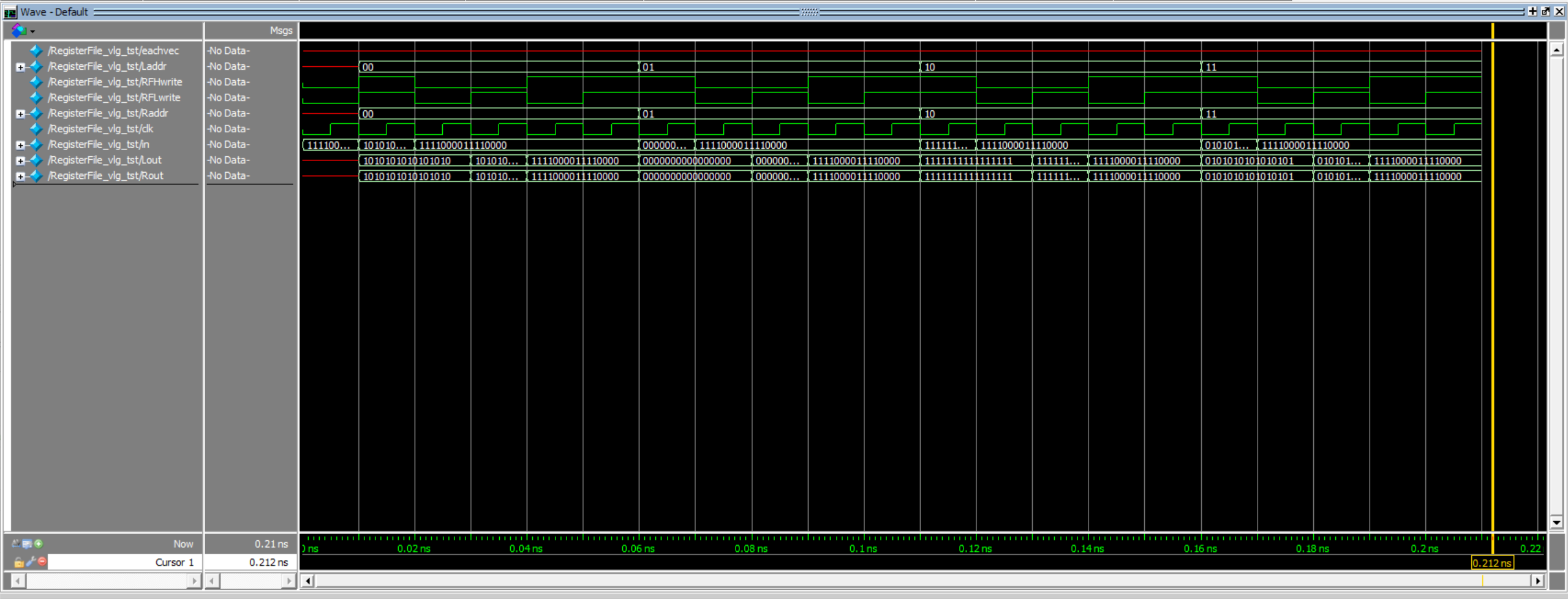
1. verilog关键代码（带注释）



1. testbench测试文件



3、仿真结果截图（带文字分析）



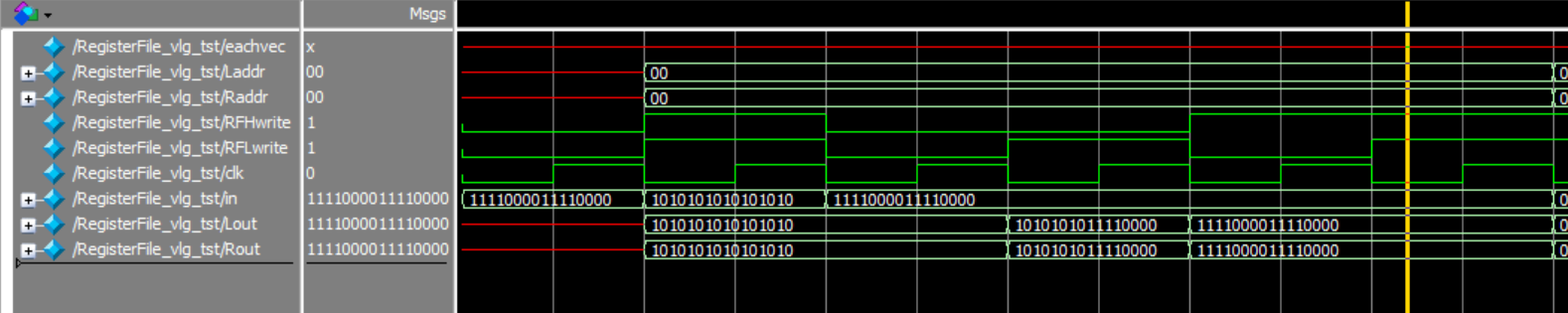
以上为仿真结果的截图，下面开始分析仿真结果.

1. 仿真前初始化



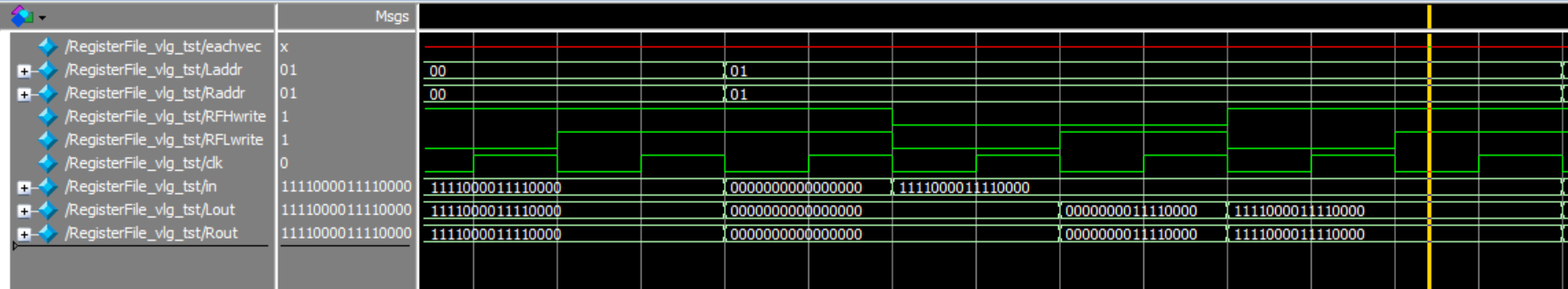
在初始化阶段，我们初始化了输入和时钟线的翻转.

1. Laddr和Raddr均为00时



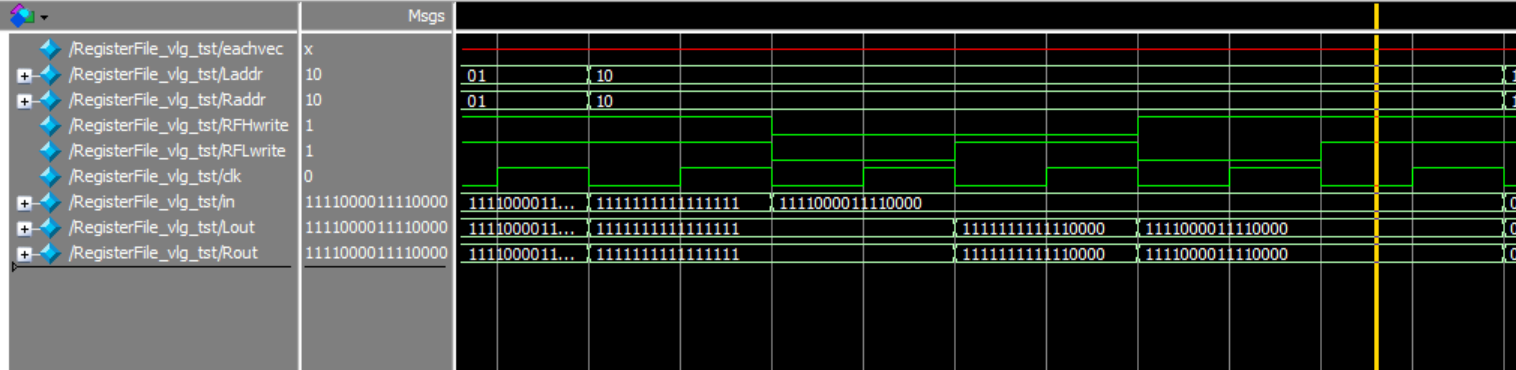
由图可知，Lout和Rout分别正确的输出了Laddr和Raddr指代的地址的数据.

1. Laddr和Raddr均为01时



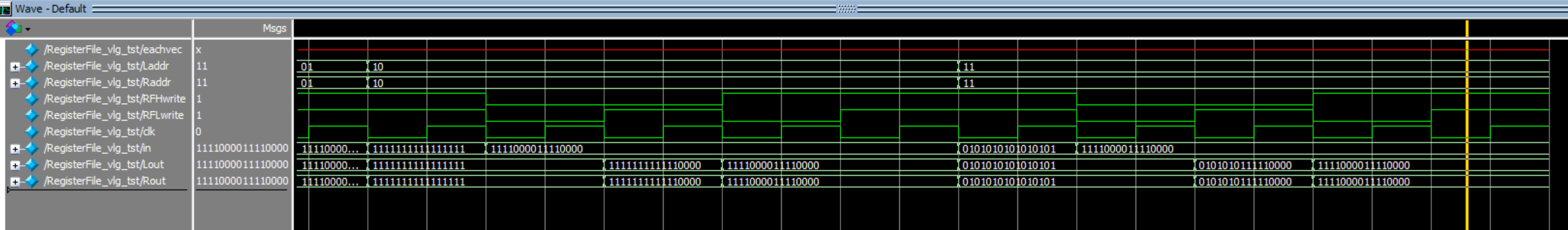
由图可知，Lout和Rout分别正确的输出了Laddr和Raddr指代的地址的数据.

1. Laddr和Raddr均为10时



由图可知，Lout和Rout分别正确的输出了Laddr和Raddr指代的地址的数据.

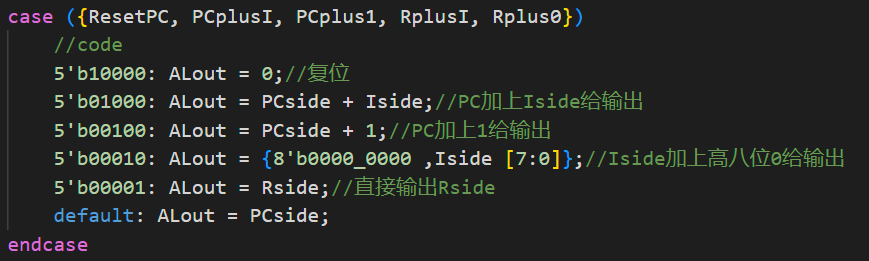
1. Laddr和Raddr均为11时



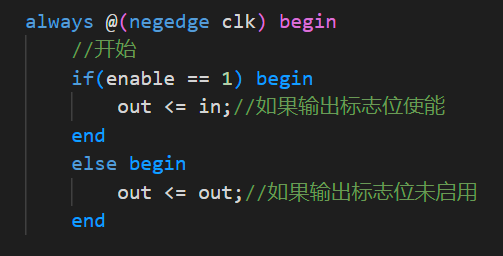
由图可知，Lout和Rout分别正确的输出了Laddr和Raddr指代的地址的数据.

寻址单元：

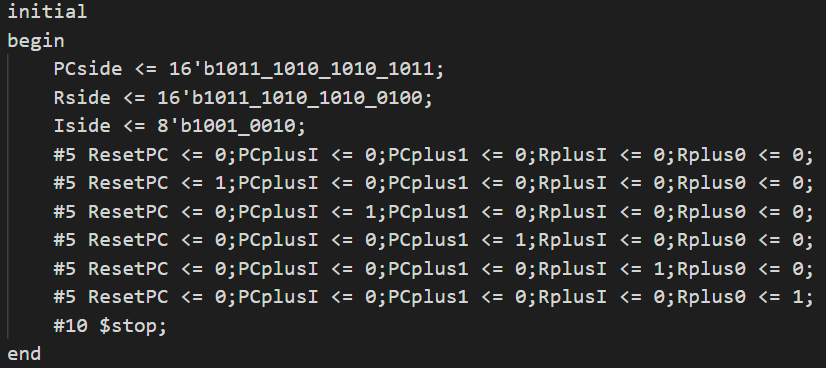
1. verilog关键代码（带注释）
   1. 此处为AddressLogic.v代码



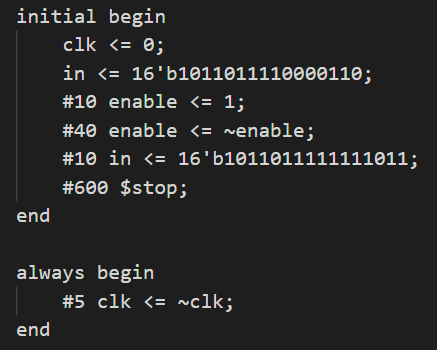
* 1. 此处为ProgramCounter.v代码



1. testbench测试文件
   1. 此处为AddressLogic.vt代码

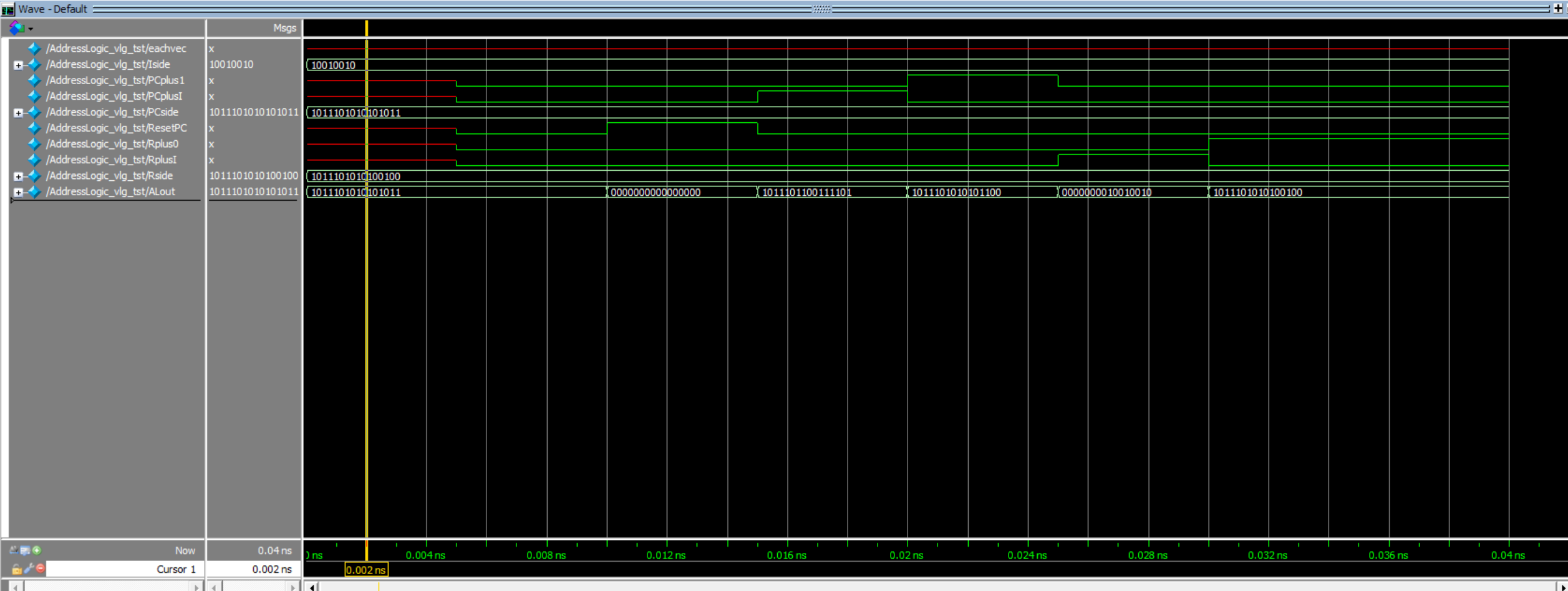


* 1. 此处为ProgramCounter.vt代码



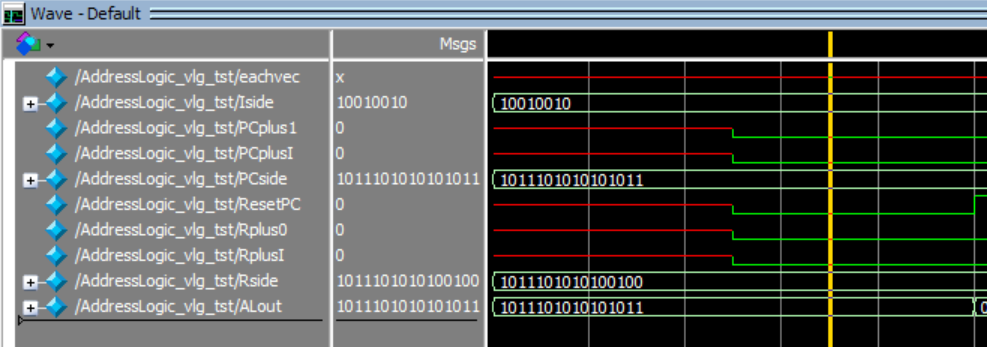
3、仿真结果截图（带文字分析）

* 1. AddressLogic.v仿真



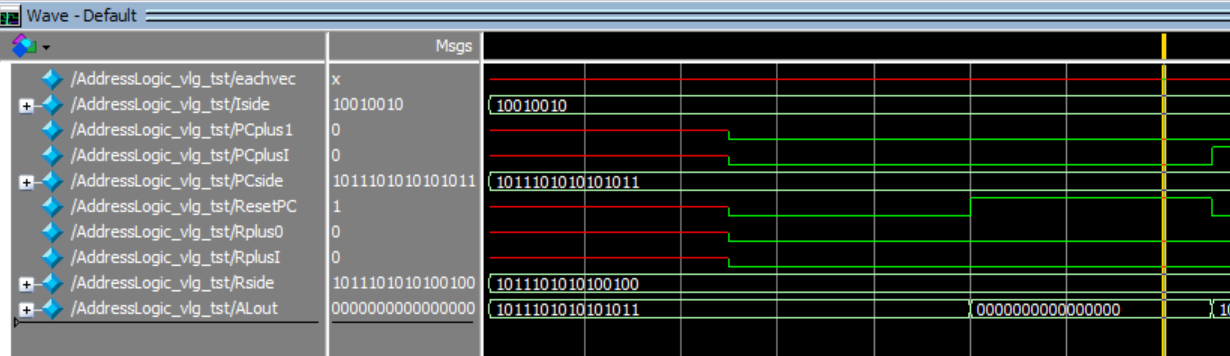
以上为仿真结果总览，下面开始仿真分析.

1. 仿真初始化



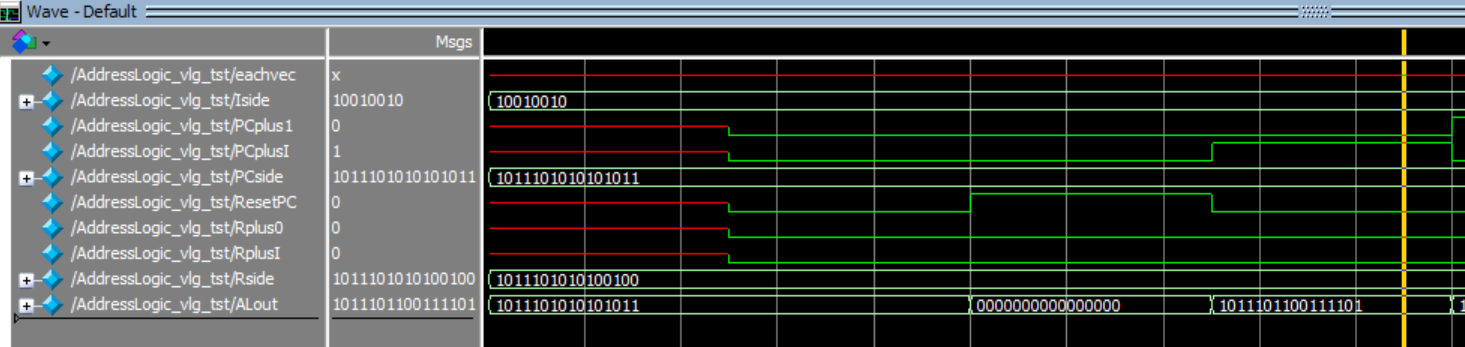
此时为仿真初始化输入模式，一切正常，全部模式置零.

1. ResetPC模式



可以看到ALout的输出被全部置零了，所以模式正确.

1. PCplusI模式



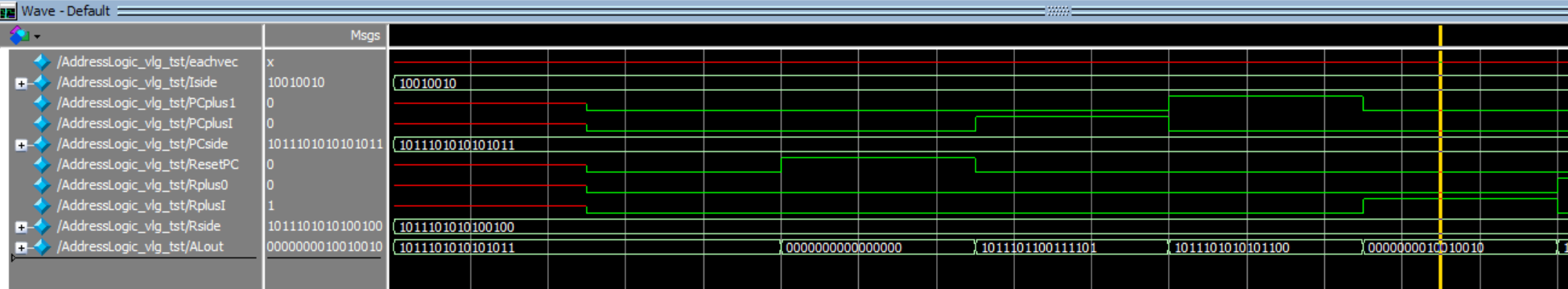
可以看到PCside和Iside正确相加了，并将结果输出给了ALout，所以结果符合预期.

1. PCplus1模式



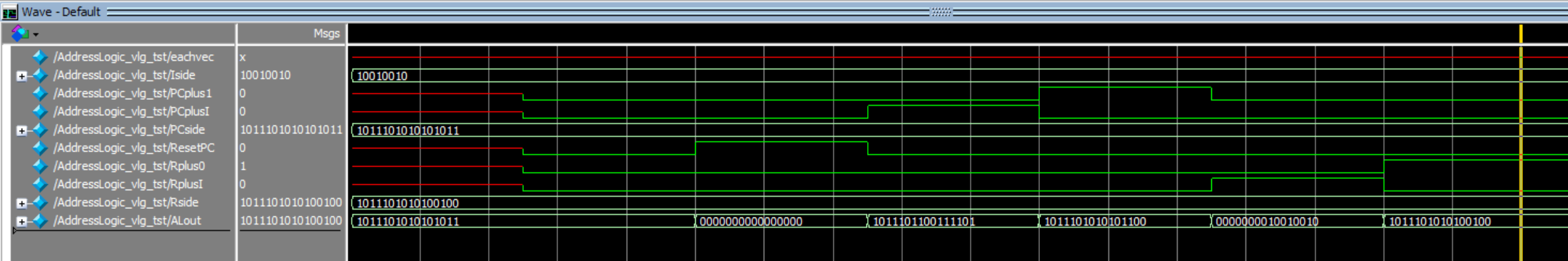
可以看到PCside + 1后的结果被输送给了ALout，所以结果符合预期.

1. RplusI模式



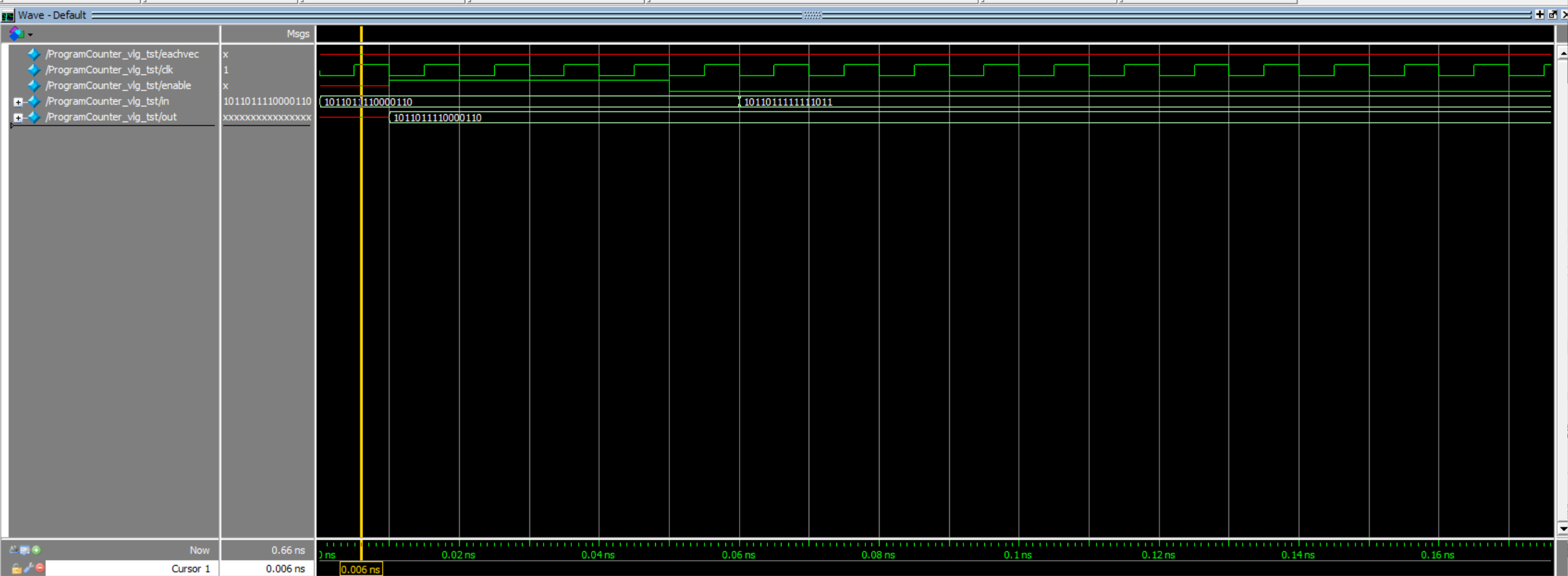
可以看到Rside和Iside正确相加了，并将结果输出给了ALout，所以结果符合预期.

1. Rplus0模式



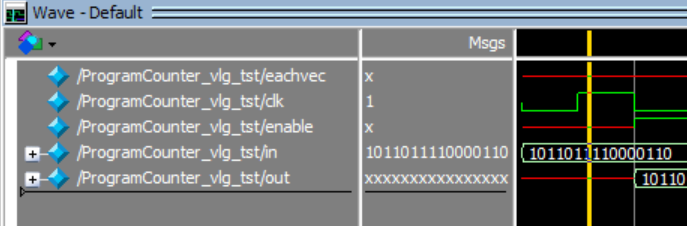
可以看到Rside + 1后的结果被输送给了ALout，所以结果符合预期.

* 1. ProgramCounter.v仿真



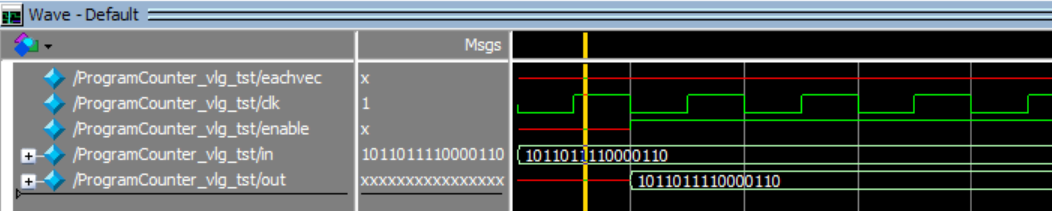
以上为仿真结果的截图总览，下面开始具体分析.

1. 仿真初始化



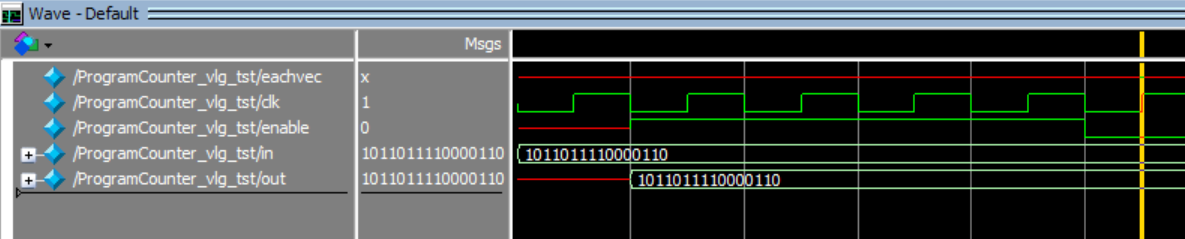
可以看到，我们初始化了输入信号和时钟线.

1. 输入使能



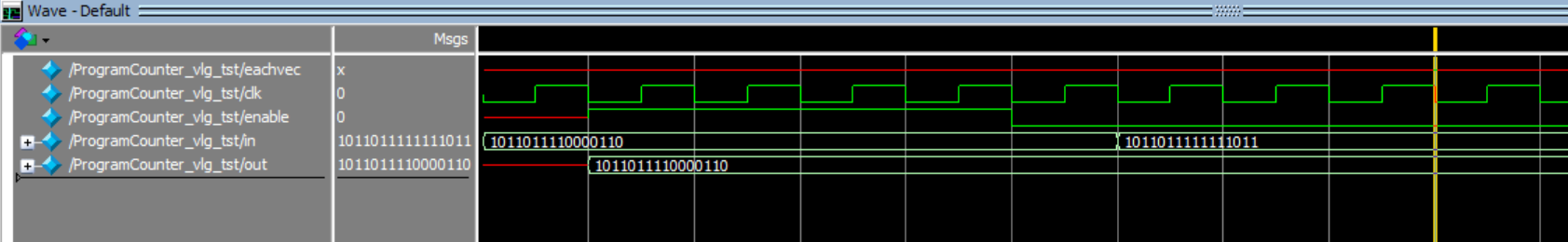
可以看到，当enable被使能时，输出立刻变成当前的输入，所以符合仿真预期.

1. 输入失能



现在我们将输入端失能.

1. 改变输入

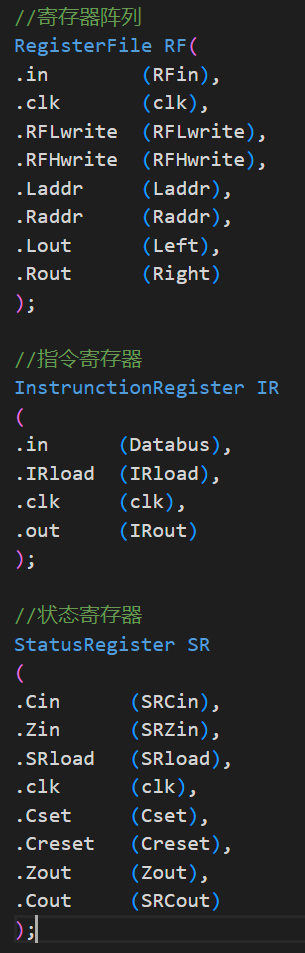
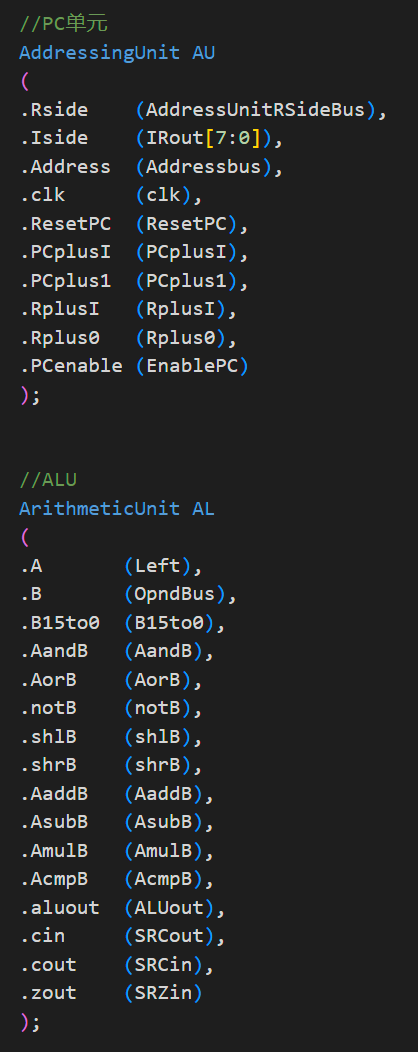


现在我们改变输入，输出不进行改变，说明功能正常.仿真结果符合预期.

**任务2**

数据通路：

1. verilog关键代码



2、关键代码分析（简述代码设计思路）



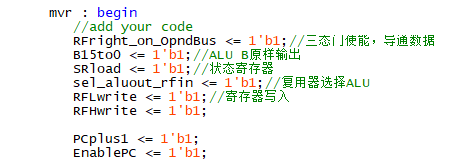
根据数据通路的数据流向图，我们可以得出各个例化模块之间的连线.

所以可以按照图表写出各个模块之间例化的连线和赋值.

**任务3**

控制通路：

1. verilog关键代码（带注释）

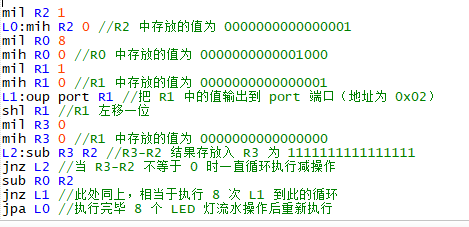


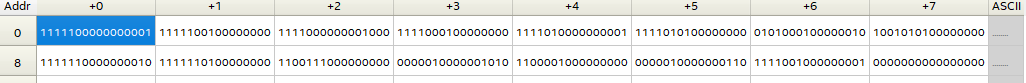
2、关键代码分析（简述代码设计思路）

首先设置三态门，将外部输入放入ALU当中，然后让ALU原样输出。再控制寄存器模块写入相应的值，实现将外部输入放入寄存器的功能.

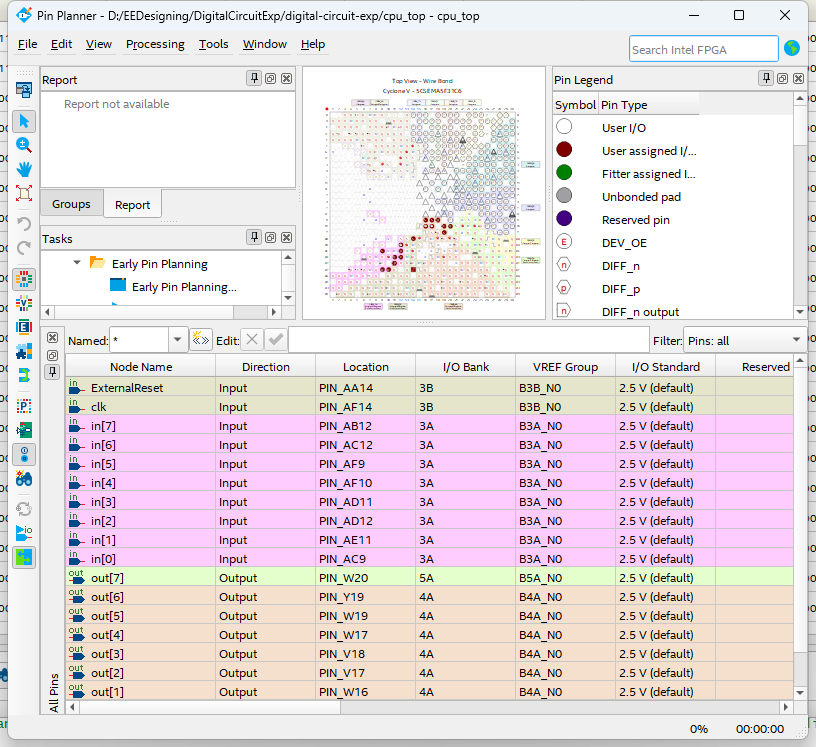
**任务4**

1. 汇编代码对应的机器码





2、下板引脚绑定



## 七、自主设计部分（选做，可加分）

**目标：**

**任务计划：**

**仿真：**

**实验现象：**

## 八、实验中遇到的问题及解决方法

1、PC程序计数器中，只读地址被误写导致寄存器工作不支持，机器码程序不能正确运行。解决办法：修改回来就行。

2、下板时不执行程序。解决办法:修改了mif文件后，需要重新配置RAM并且编译。

## 九、思考题

## 该cpu执行一条指令需要几个时钟周期？每个时钟周期分别完成那些功能？

一般需要4个时钟周期，每个时钟周期完成：取得数据、读取寄存器，运算，输出和保存。

**该cpu寄存器阵列共有多少个寄存器？如何用指令给这些寄存器赋值？**

4个寄存器，使用mil和mih指令分布对高低8位执行赋值。

**如果没有标志位寄存器，会对cpu哪些功能造成什么影响？**

运算功能，比较功能。ALU不能完成负数补码的运算，实现比较跳转的功能。

## 十、实验分工说明

邓皓宇：控制通路，CPU整合部分报告撰写，完成汇编流水灯下板实验

熊仔羿：数据通路及其整合，完成汇编流水灯下板实验

## 十一、实验感想和建议

这门课程让我更深入地了解了计算机系统的工作原理，并且让我学会了使用verilog语言来描述计算机系统中的电路。详细地全流程地编写调试了一个简易的CPU，知道了如何从仿真波形中看出逻辑错误，从报错中找到语法错误。加深了对数字电路、数字设计的了解。通过完成这门课程的实验，我感受到了计算机硬件设计的乐趣，并且提升了我的动手能力和技术水平。

建议：老师发的代码可以再整理规范详细一点。指导书编排感觉有点乱。