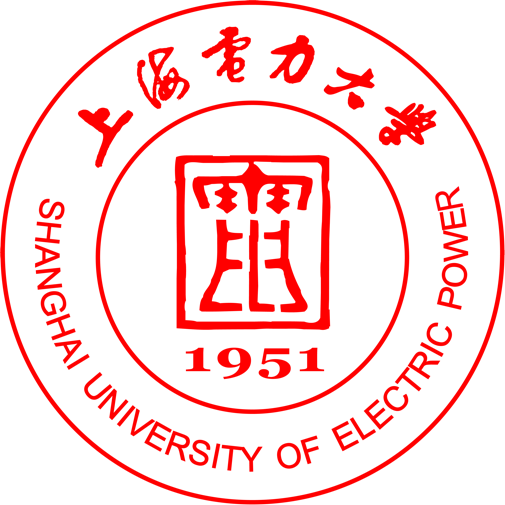
**上 海 电 力 大 学**

《数字集成电路设计与分析》实验报告



**实验题目：**  VCS仿真调试基础实验

**专 业：**

**班 级**  **学号**

**姓 名**

**时 间**  2023-10-18

1. 实验目的

通过使用VCS编译一个已存在的verilog设计进行VCS可执行仿真

通过执行由VCS产生的二进制可执行仿真对Verilog设计的operation进行仿真

通过阅读由Verilog源代码中的Verilog系统任务调用产生的控制台信息决定Verilog设计是否通过了验证了

使用Verilog系统任务调用调试一个已存在的Verilog设计

VCS CLI调试器调试

1. 实验任务及要求

内容：VCS实验指导书 实验1、实验2 A和B部分

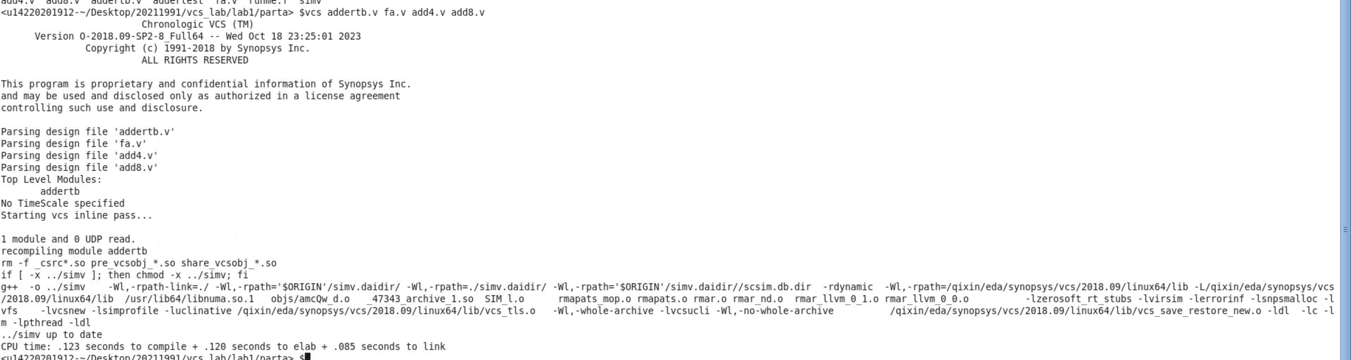
1. 实验内容及步骤

实验1

A部分：两步骤仿真过程

任务1 编译并产生可执行仿真文件

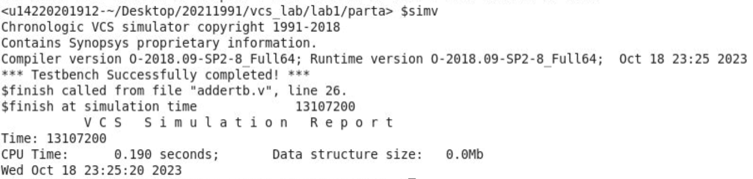
vcs addertb.v fa.v add4.v add8.v



任务2 运行仿真

运行测试台文件并执行simv对设计进行仿真。

simv



任务4 使用不同的名称创建可执行仿真

vcs addertb.v fa.v add4.v add8.v –o addertest

addertest



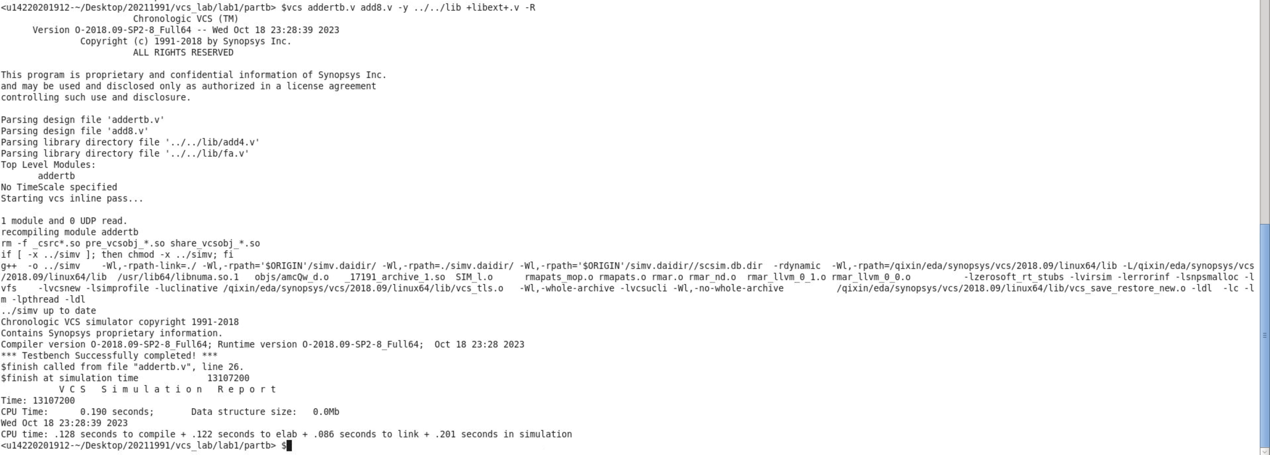
B部分 使用库目录

任务1 使用设计库目录进行编译和仿真

cd ../partb

ls

vcs addertb.v add8.v –y ../../lib +libext+.v -R



任务2 使用-f参数进行编译

实验2 VCS调试基础

A部分：使用Verilog系统任务调用进行调试

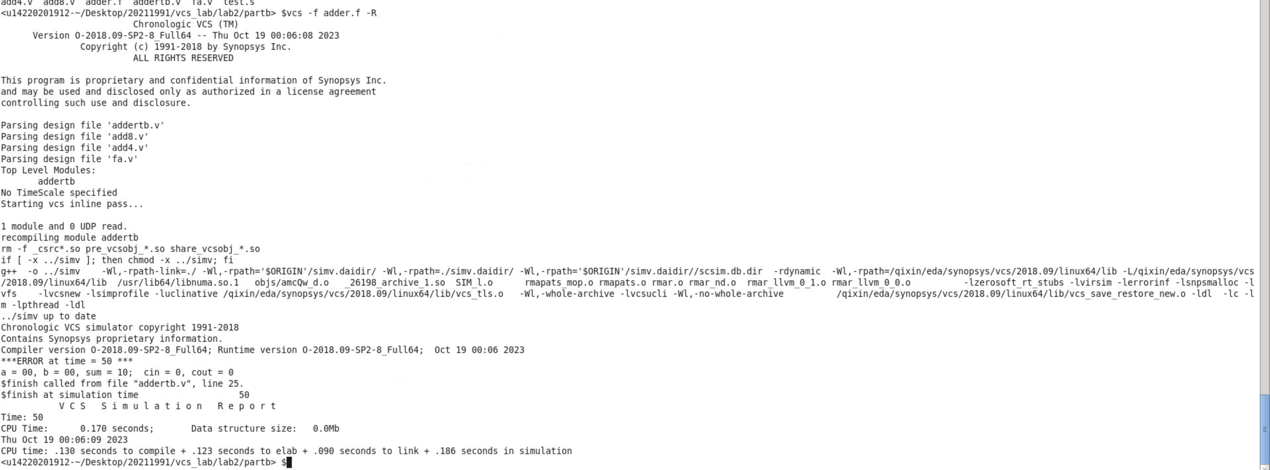
任务1 有用的VCS参数

cd ../../lab2/parta

vcs -h

任务2 编译并运行第一遍验证，对carry select加法器进行编译并仿真。

vcs –f adder.f -R



任务3 调试错误

将$finish改为$stop

通过这种调整，每次发现错误，$display列出的结果都将被打印输出，测试台将在这个时间点中断。

下面是经过修改的代码

module addertb.v;

reg [7:0] a\_test,b\_test;

wire [7:0] sum\_test;

reg cin\_test;

wire cout\_test;

reg [17:0] test;

add8 u1(a\_test,b\_test,cin\_test,sum\_test,cout\_test);

initial

if (!$test$plusargs("monitoroff"))

$monitor ($time, " %h+%h=%h;cin=%h,cout=%h",

a\_test,b\_test,sum\_test,cin\_test,cout\_test);

initial

begin

for (test=0,test<=18'h1ffff; test=test+1) begin

cin\_test=test[16];

a\_test=test[15:8];

b\_test=test[7:0];

#100;

if ({cout\_test,sum\_test}!= =(a\_test+b\_test+cin\_test)) begin

$display("\*\*\*ERROR at time = %0d \*\*\*",$time);

$display("a=%h,b=%h,sum=%h,cin=%h,cout=%h",

a\_test,b\_test,sum\_test,cin\_test,cout\_test);

$display("\nIn add4(u1)");

$display("a=%b,b=%b,sum=%b,cin=%b,c=%b,cout=%b",

u1.u1.a,u1.u1.b,u1.u1.sum,u1.u1.cin,u1.u1.c,u1.u1.cout);

$display("\nIn add8(u1)");

$display("a=%b,b=%b,sum=%b,cin=%b,cout=%b",

u1.a,u1.b,u1.sum,u1.cin,u1.cout);

$finish;

end

end

$display("\*\*\*Testbench Successfully completed!\*\*\*");

$finish;

end

endmodule

B部分 使用VCS CLI调试器进行调试

任务1 编译并运行第一遍验证

cd ../partb

vcs –f adder.f -R

任务2 使能CLI调试器参数进行编译，修改`$finish`为`$stop`，并运行

任务3 使用CLI调试器进行调试

查看信号的值，使用`show 信号 -value -radix hex/bin/dec`命令,显示信号的值，以特定的进制显示。

run

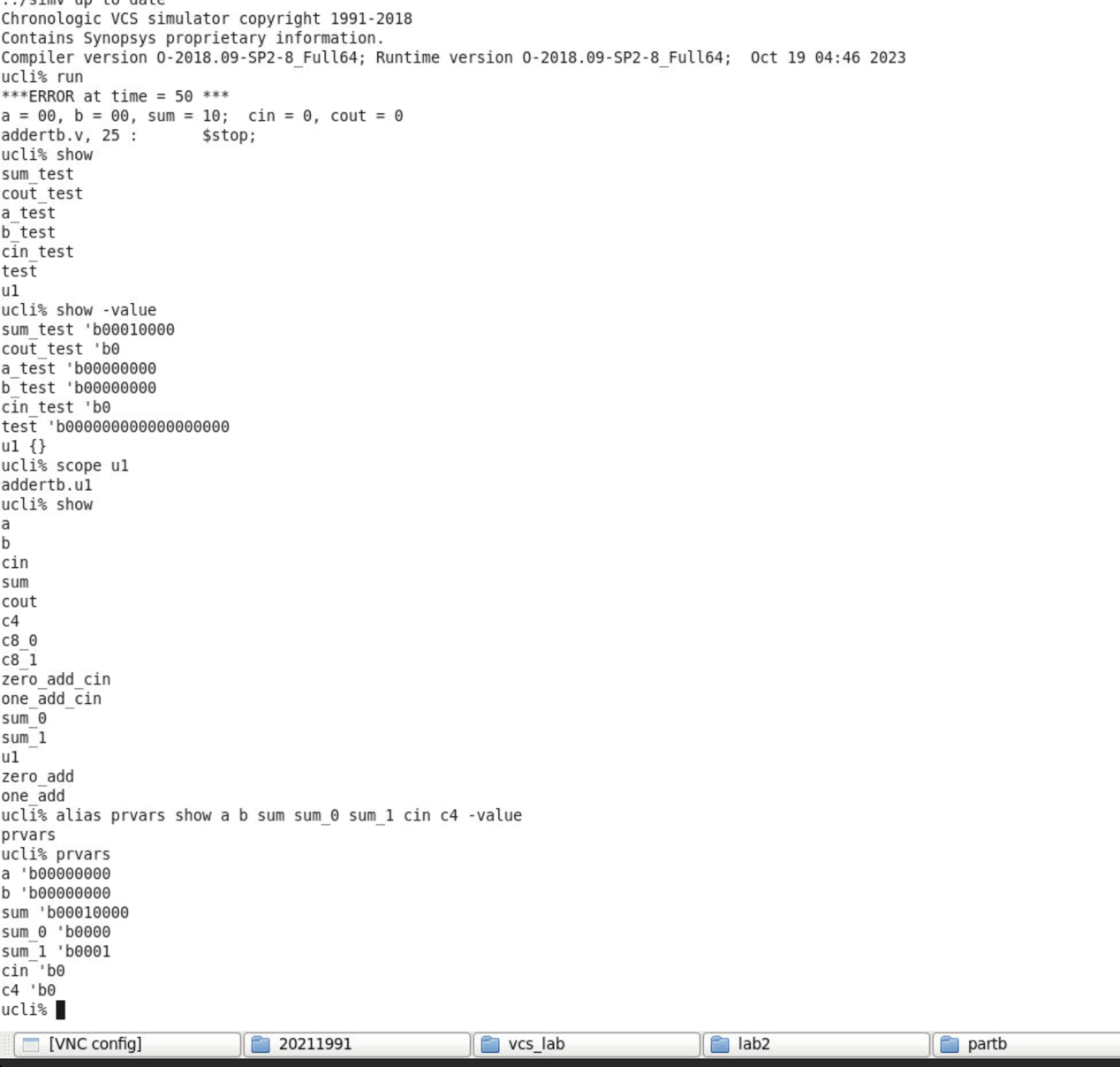
show -value # 等价于 show a\_test b\_test cin\_test cout\_test sum\_test -value

scope u1

show

在调试过程中，你常需要多次输出变量相同设置的值。为这个经常被执行的任务创建一个别名会让工作变得简单一些。一旦创建了别名，执行时只需要输入别名就可以了。

alias prvars show a b sum sum\_0 sum\_1 cin c4 -value

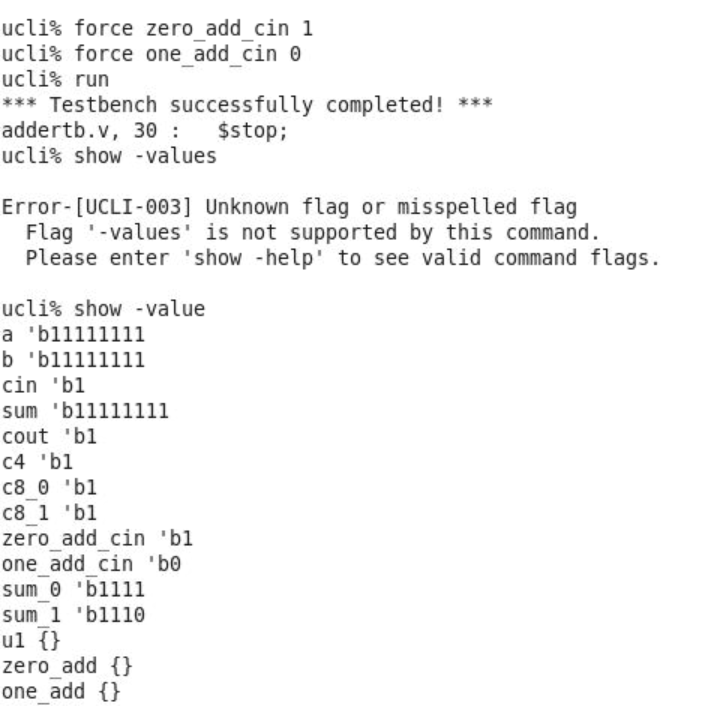
prvars

2:1选择器应该sum\_0为0的值但是却选择了sum\_1为1的值。

将sum\_0加法器的carry-in改变为“1”并将sum\_1加法器的carry-in改变为“0”，问题就得到有效的改正了。

force zero\_add\_cin 1

force one\_add\_cin 0



1. 实验总结