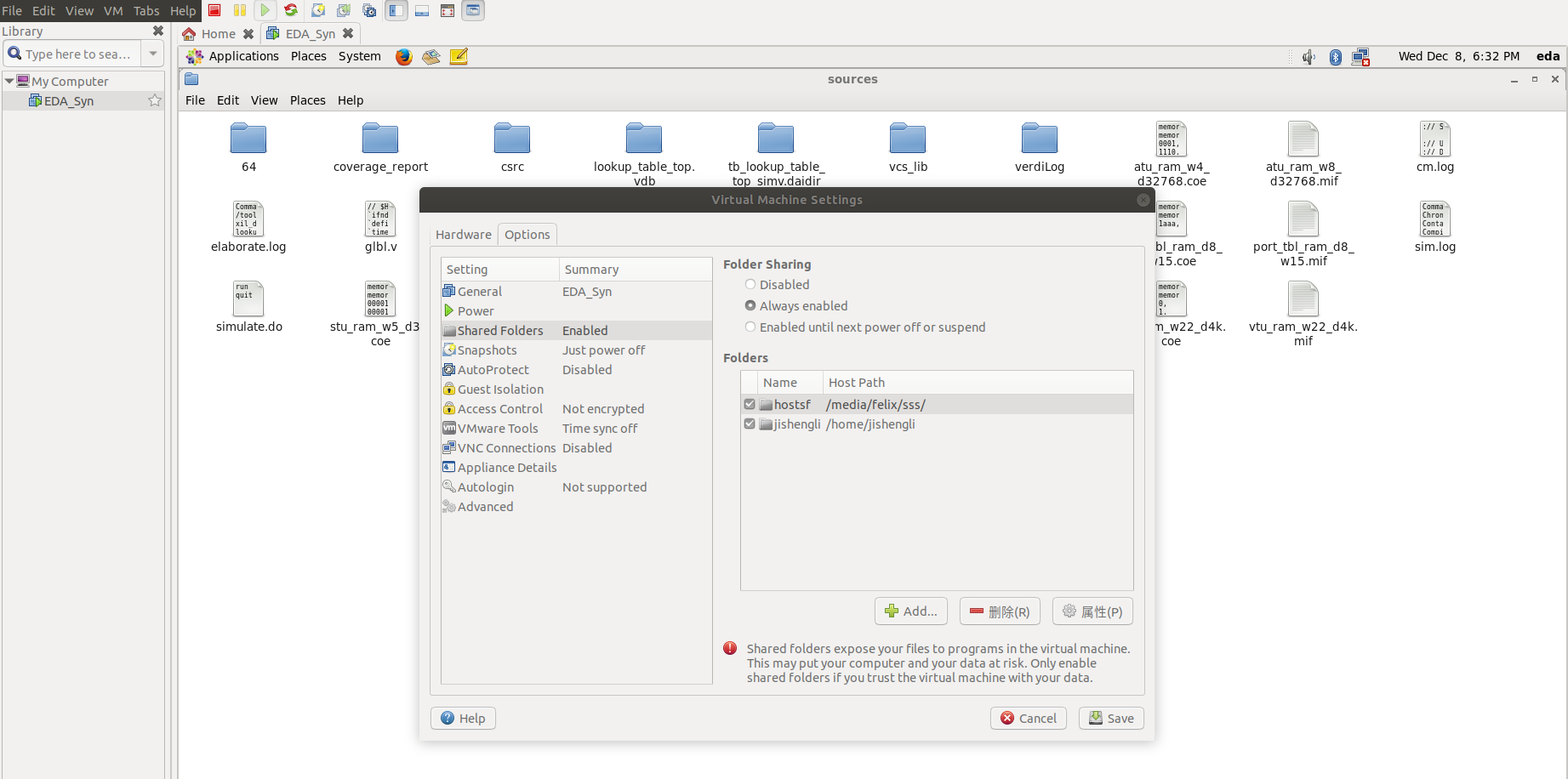
VCS+VERDI独立仿真xilinx工程说明

使用环境：ubutun系统+centos7虚拟机

第一步：添加centos7虚拟机和ubutun系统共同的开发空间。将ubutun系统下的工程目录添加到centos7虚拟机中，如下图所示：

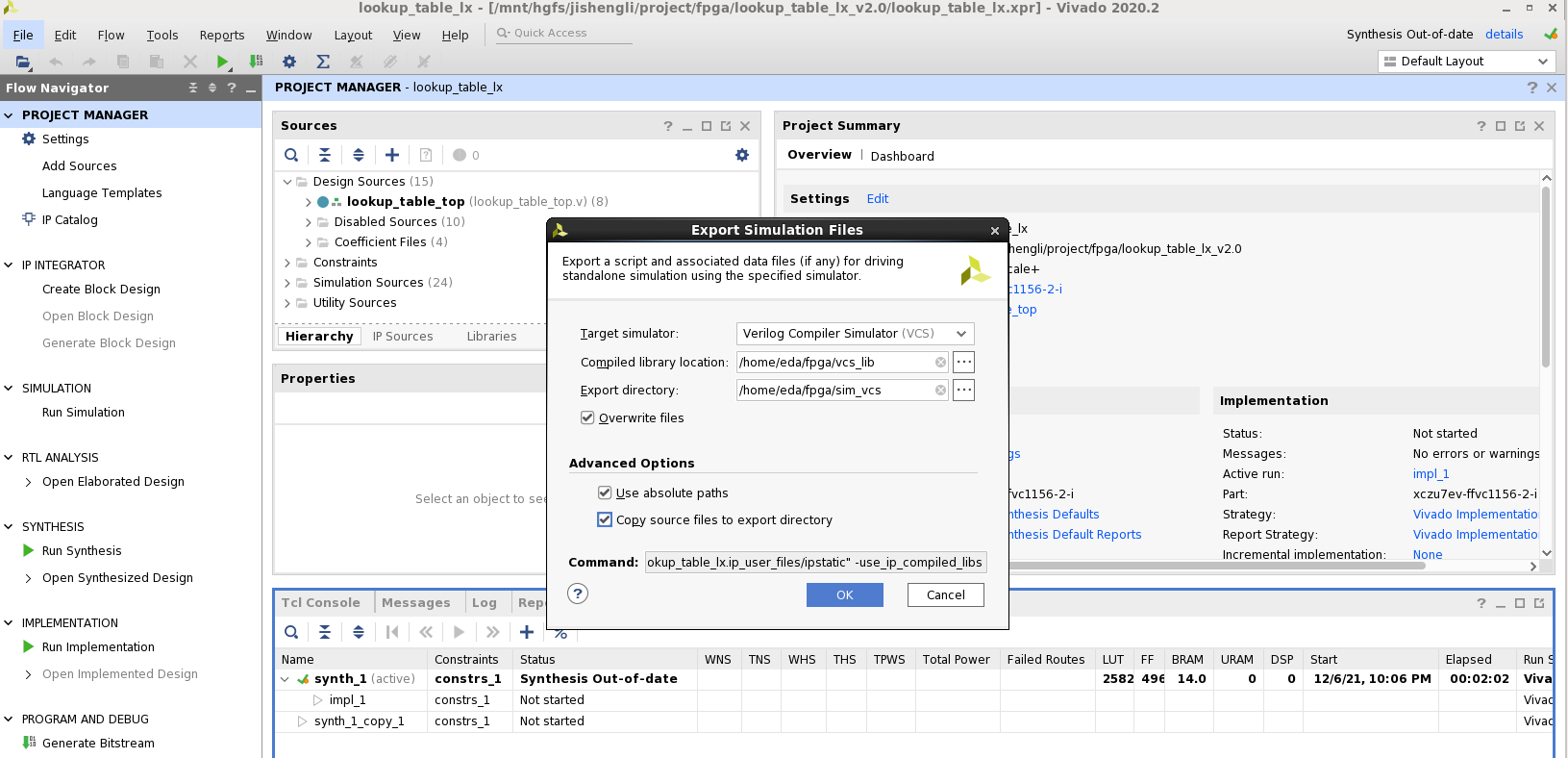
虚拟机：VM-->setting-->option，添加在ubutun系统下的工程目录。

第二步：在centos7系统下启动vivado，并打开share工作区的工程 ，然后导出simulation文件。

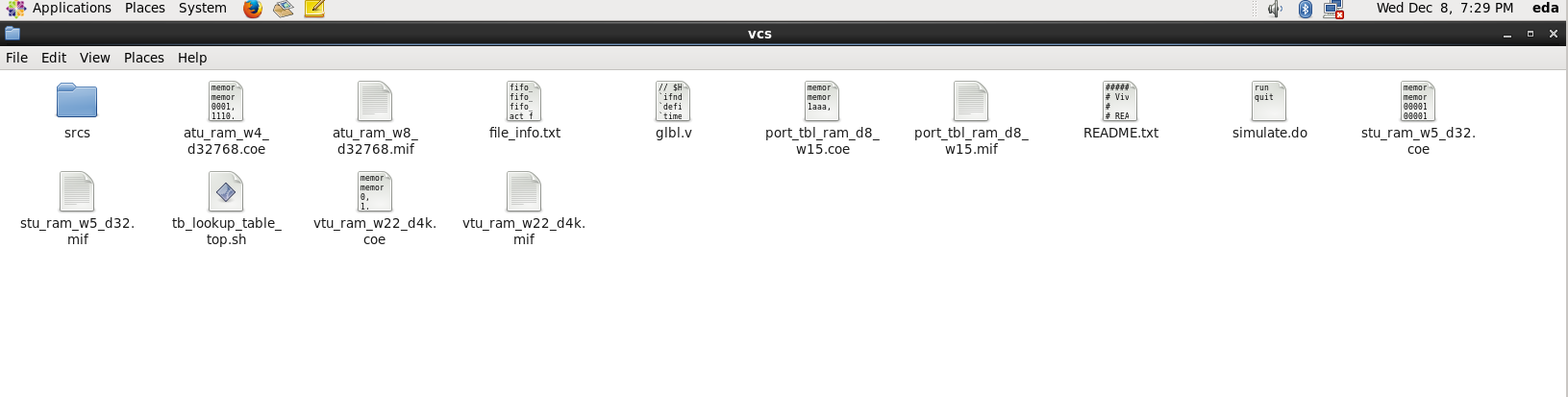
(share工作区目录：/mnt/hgfs/jishengli)

vivado: file-----> export----> export simulation

然后可以关掉vivado工程了。



第三步：打开导入simulation文件的文件夹，里面包含一个vcs文件夹。打开如下图所示。



在VCS文件夹下新建一个makefile文件和synopsys\_sim.setup文件。

1）：synopsys\_sim.setup中填写内容如下：

xil\_defaultlib:vcs\_lib/xil\_defaultlib

OTHERS=/home/eda/fpga/vcs\_lib/synopsys\_sim.setup

2）：Makefile文件中添加内容如下：（标红和标黄的地方是需要根据具体工程修改的地方）。

其中ref\_dir=/home/eda/fpga/sim\_vcs/vcs/srcs可从当前目录下的后缀为.sh(tb\_lookup\_table\_top.sh)文件中复制（需要删除双引号），表示源代码的位置；

top是代码顶层的名称；黄色部分为testbench文件的名称。

#-------------------------------------------------------------------------------------------------------

all:comp elaborate sim

#-------------------------------------------------------------------------------------------------------

vlogan\_opts=-full64 -v2005

vhdlan\_opts=-full64

vcs\_elab\_opts=-full64 -debug\_pp -t ps -licqueue -l elaborate.log -P ${LD\_LIBRARY\_PATH}/novas.tab ${LD\_LIBRARY\_PATH}/pli.a

vcs\_sim\_opts=-ucli -licqueue -l sim.log

top=lookup\_table\_top

ref\_dir=/home/eda/fpga/sim\_vcs/vcs/srcs

#-------------------------------------------------------------------------------------------------------

comp :

内容参考3）

#-------------------------------------------------------------------------------------------------------

#-------------------------------------------------------------------------------------------------------

elaborate:

vcs ${vcs\_elab\_opts} xil\_defaultlib.tb\_${top} xil\_defaultlib.glbl -o tb\_${top}\_simv \

-cm line+cond+fsm+branch+tgl \

-cm\_name ${top} \

-cm\_dir ./${top}.vdb

#-------------------------------------------------------------------------------------------------------

sim:

./tb\_${top}\_simv ${vcs\_sim\_opts} -do simulate.do \

-cm line+cond+fsm+branch+tgl \

-cm\_name ${top} \

-cm\_dir ./${top}.vdb

#-------------------------------------------------------------------------------------------------------

coverage:

dve -full64 -covdir \*.vdb

#-------------------------------------------------------------------------------------------------------

urg:

urg -full64 -dir ${top}.vdb -report coverage\_report &

#-------------------------------------------------------------------------------------------------------

verdi:

verdi +v2k -f filelist.f -f -ssf \*.fsdb -nologo &

#-------------------------------------------------------------------------------------------------------

clean:

rm -rf \*~ core csrc \*simv vc\_hdrs.h ucli.key urg\* \*.log novas.\* \

\*.fsdb\* verdiLog 64\* DVEfiles \*.vpd \*\_report \*.vdb \*.daidir

#-------------------------------------------------------------------------------------------------------

3）：打开VCS文件夹下的.sh文件（例如tb\_lookup\_table\_top.sh）如下图所示。复制compile()下面的大括号{}内所有内容。然后打开makefile文件。将复制的内容粘贴到makefile文件中comp：下方。

如下第二张图所示。

因为makefile和shell语法的不同，需要对粘贴的内容作如下更改：

$vlogan\_opts需要替换为${vlogan\_opts}

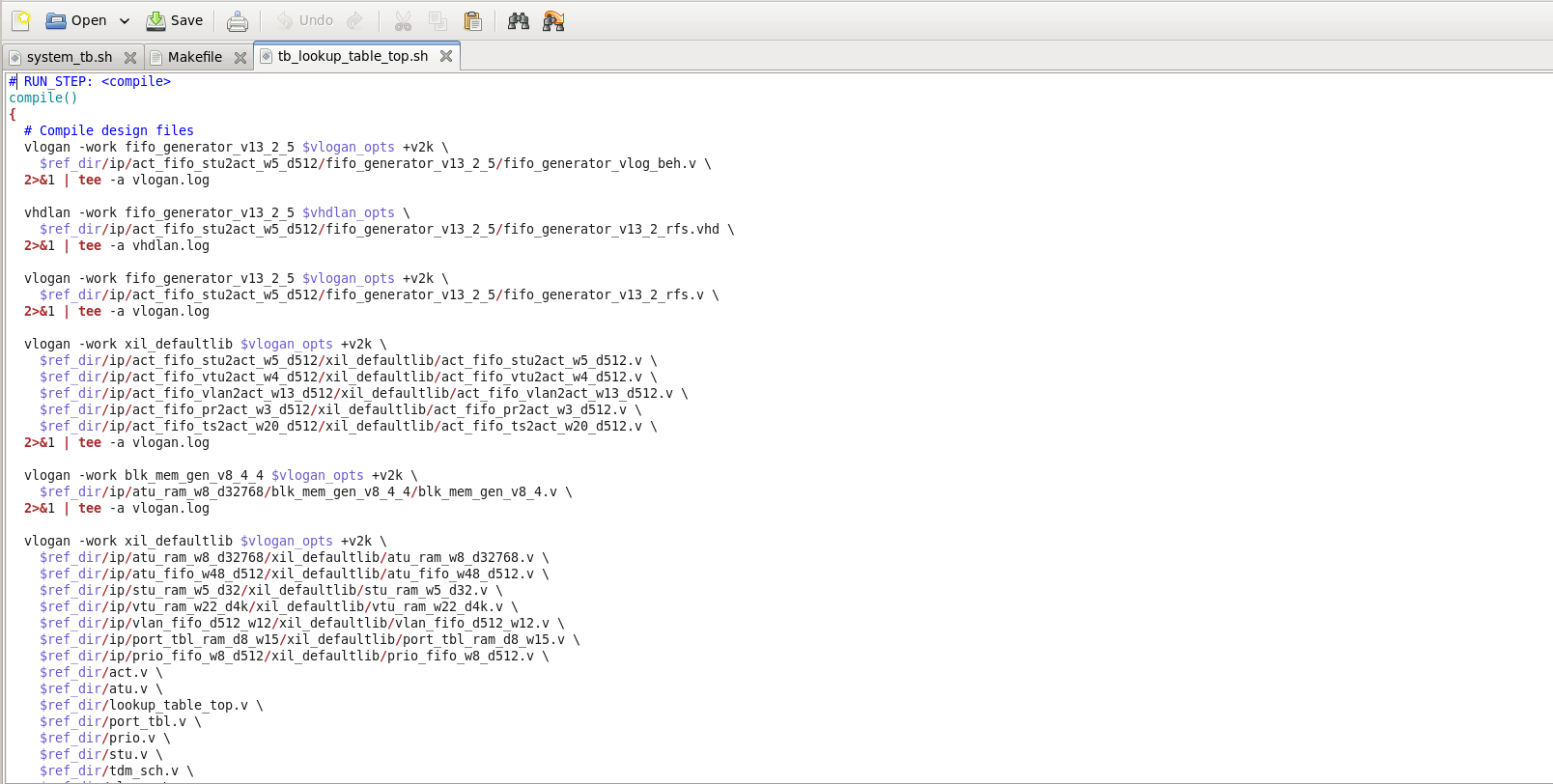
$ref\_dir需要替换为${ref\_dir}

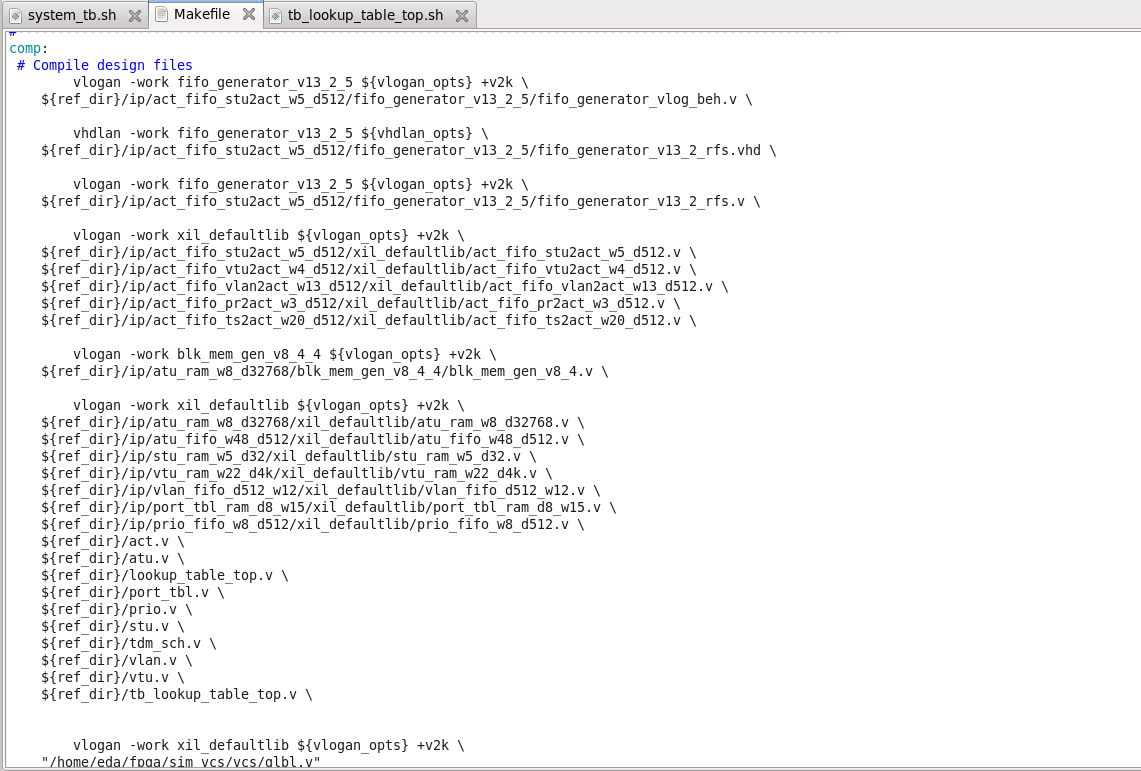
$vhdlan\_opts需要替换为${vhdlan\_opts}

删除所有“2>&1 | tee -a vlogan.log”，“2>&1 | tee -a vhdlan.log”语句。

vlogan和vhdlan等指令需要以tab键开头,需要将对应行开头的空格替换成tab。

该部分最后一行不能有换行符\，有的话删除即可。





4）：在src文件夹下，顶层的tb文件中加入生成fsdb文件的语句

标黄部分需要改成对应的顶层和testbench文件名称。

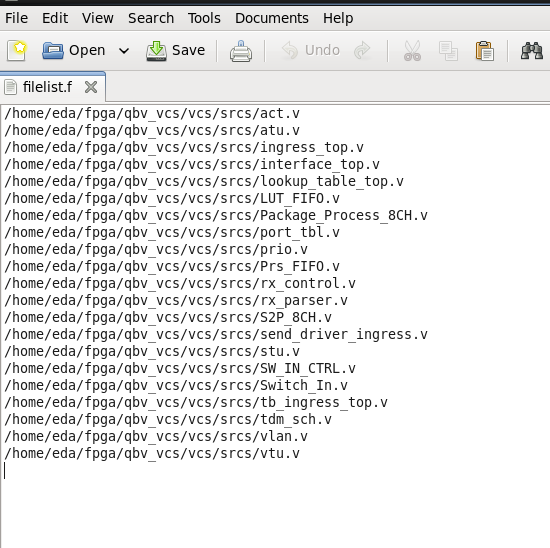
initial begin

$fsdbDumpfile("lut\_top.fsdb");

$fsdbDumpvars(0,tb\_lookup\_table\_top);

end

5）在VCS文件夹下新建一个filelist.f文件，并将VCS文件夹下SRCS子文件夹下的.v文件加入到filelist.f文件中。如果想看IP内部的信号，需要将IP文件夹内的.v文件一并加入即可。如下图所示。



6）：其中make comp表示语法检查；make elaborate 表示编译；make sim 表示运行仿真；make coverage表示观察覆盖率；make urg表示产生覆盖率报告；make verdi表示运行verdi软件查看仿真波形；make clean表示清除所有产生的文件。make表示顺序执行comp，elaborate，sim三个语句。

VCS和vivado有部分差异，在VCS中报错的部分需要按照提示修改，直到通过编译。

调试脚本的时候可分步执行。调试代码可直接make，然后make verdi来仿真。

7）：编译过程中，可通过在testbench中增加$finish语句结束，也可以通过修改VCS文件夹内的simulate.do文件来约束运行时间，如下所示。

