Verdi基础

- 查看设计
- **debug**
- **Verdi不能自己产生波形**

1.Verdi环境配置

.bashrc中配置

```
export Verdi_HOME=$Synopsys_Dir/Verdi2015
#export NOVAS_HOME=$Synopsys_Dir/Verdi2015
export PATH=$Verdi_HOME/bin:$PATH
export LD_LIBRARY_PATH="/opt/Synopsys/Verdi2015/share/PLI/lib/linux64"$PATH
export LD_LIBRARY_PATH="/opt/Synopsys/Verdi2015/share/PLI/VCS/linux64"$PAT
```

2.VCS产生Verdi波形

- tb中加入相应的系统函数

异步FIFO设计

```
initial begin
   $fsdbDumpfile("fifo.fsdb");
   $fsdbDumpvars(0);
end
```

- makefile中加入相应的选项

```
#*************parameter input**************
#※prepare the source list file  and then make add the soucefile name
#for example ,the Verilog source file's name is vlog_list.f then :make norsim src_list=file_list
src_list =  sim
simv_name = simv
vpdpluse_name = vcdpluse
cov_file_name = coverage
vdb_name = $(simv_name)
#*************constant command*************
#compile
NOR_VCS = vcs -full64 -sverilog +v2k -timescale=1ns/1ns          \
     -debug_all                          \
     +notimingcheck                            \
     +nospecify                          \
     +vcs+flush+all                            \
     -o $(simv_name)   \
     -l compile.log  \
        -f $(src_list).f

#coverage compile switch
COV_SW = -cm line+cond+fsm+branch+tgl

#verdi dump wave compile option
VERDI_SW = -P /opt/Synopsys/Verdi2015/share/PLI/VCS/linux64/novas.tab \
     /opt/Synopsys/Verdi2015/share/PLI/VCS/linux64/pli.a

#run option
RUN_GUI = -R -gui -l run.log
RUN_VPD = -R +vpdfile+$(vpdpluse_name).vpd -l run.log
```

```
RUN_COV = -R $(COV_SW) -cm_name $(vdb_name) -cm_dir ./$(cov_file_name) -l run.log
RUN_VER = -R +fsdb+autoflush -l run.log
#************command************
#normally sim
norsim:
	$(NOR_VCS) $(RUN_GUI)

#post-process
postsim:
	$(NOR_VCS)  $(RUN_VPD)
	dve -vpd $(vpdpluse_name).vpd

#coverage
covsim:
	$(NOR_VCS) $(COV_SW) $(RUN_COV)
	dve -covdir $(cov_file_name).vdb

#verdi
versim:
	$(NOR_VCS) $(VERDI_SW) $(RUN_VER)
	verdi -sv -f $(src_list).f -ssf *.fsdb -nologo


#rm
clr:
	rm -rf *csrc ./*.daidir $(simv_name) *simv* DVE* ucli* *.vpd *.vdb *.log *.fsdb *novas* *.dat *Log
	*rc *conf
```

3.nTrace介绍

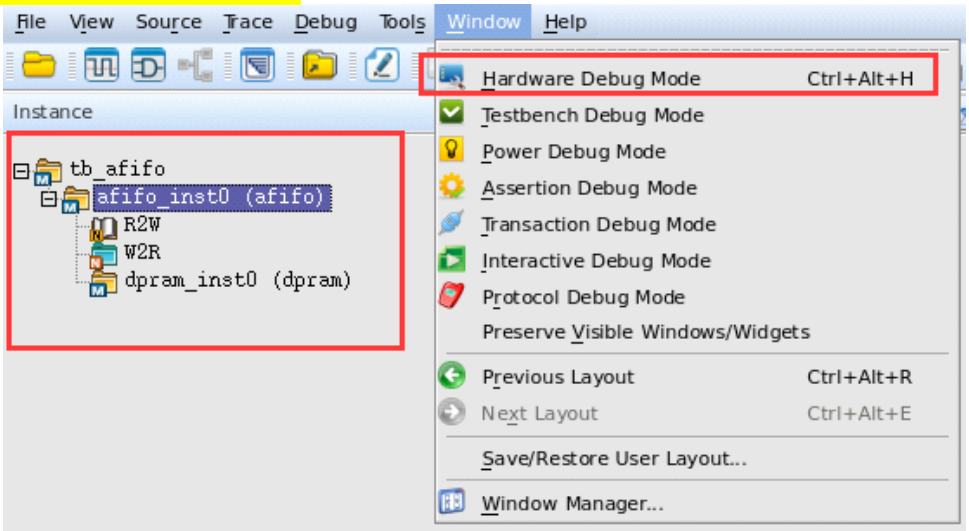
- 如何调用Verdi？

  Verdi

  verdi -nologo

  makefile中的命令

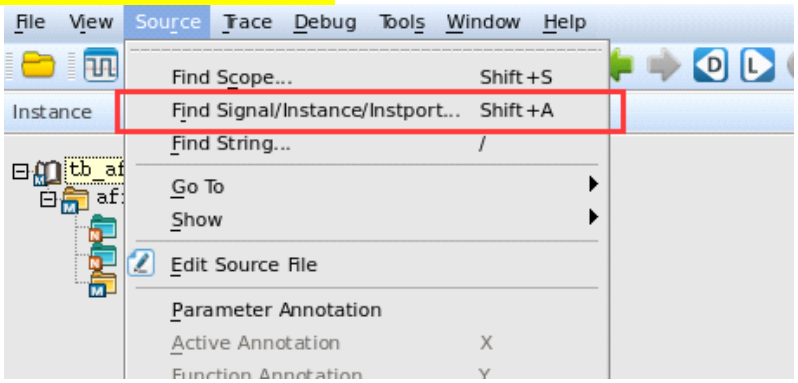
- 如何查看包含的设计架构？



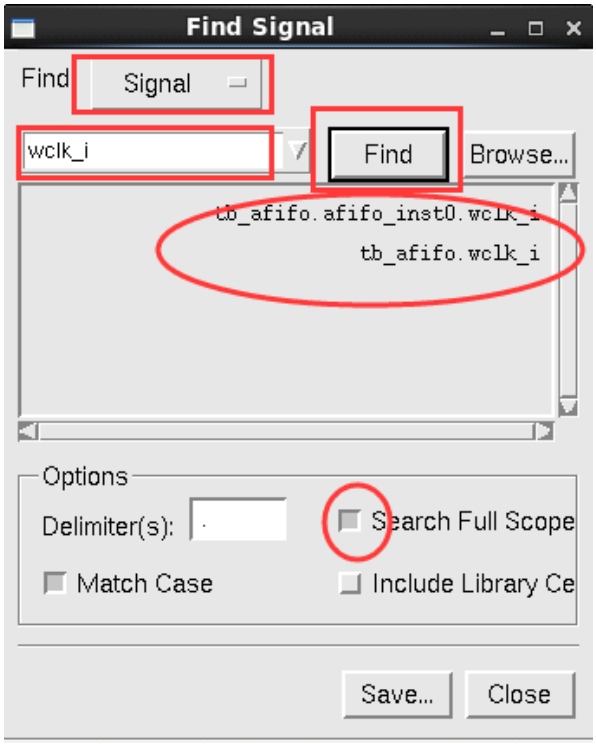
- 如何查寻模块实例化的位置？

①在设计结构中双击实例化模块的名字 ->源代码窗口高亮了模块名
②再双击高亮的模块名->看到在上层模块中的哪一行被调用了

• 在nTrace中如何查找模块和信号？

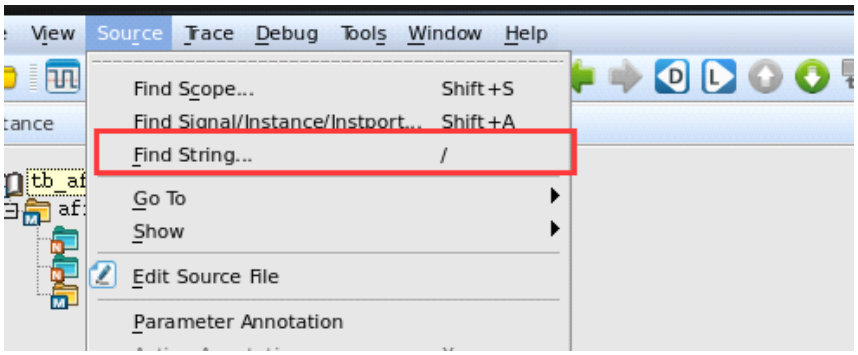


或Shift+A：

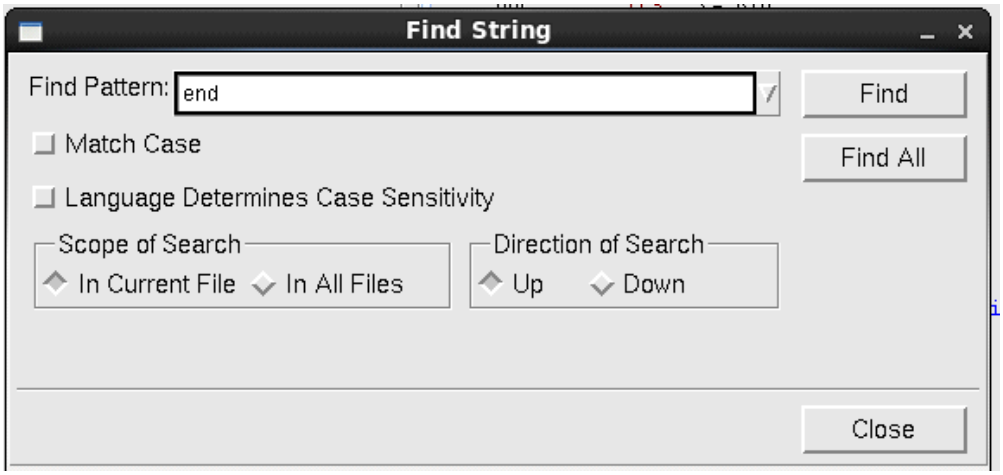


• 如何查寻字符串？

类似于文本查找功能。

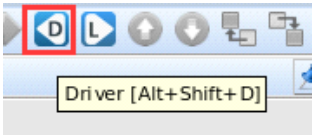


或者快捷键斜杠/：



• 如何查看某个信号是被哪些信号驱动的？

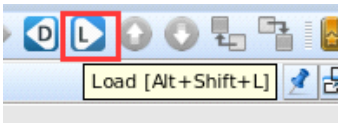
双击信号名，或者单击选中信号名然后点击：



接着可以通过快捷键N查看下一个被赋值/驱动的位置 或者快捷键P查看上一个被驱动/赋值的位置。

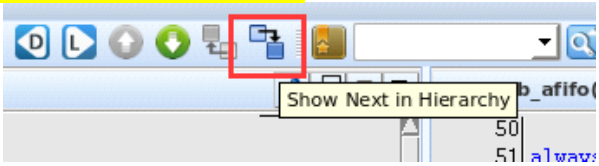
• 如果查看某个信号驱动了哪些信号？

选中信号，然后点一下：



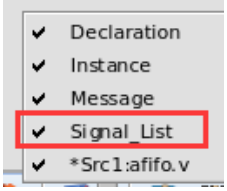
接着可以通过快捷键N查看下一个驱动其他信号的位置 或者快捷键P查看上一个驱动其他信号的位置。

• 如何查找位于不同层次的驱动信号？



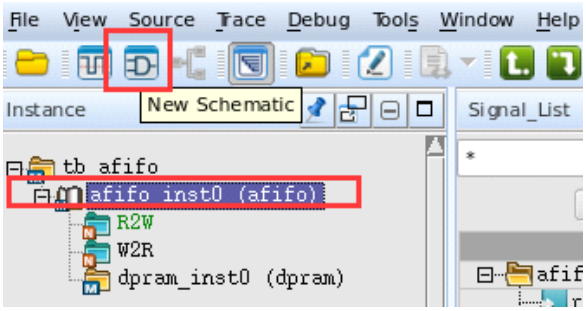
• 如何查看设计有哪些信号？

通过signal_list来查看，可以选择查看输入/输出等端口类型。



4.nSchema

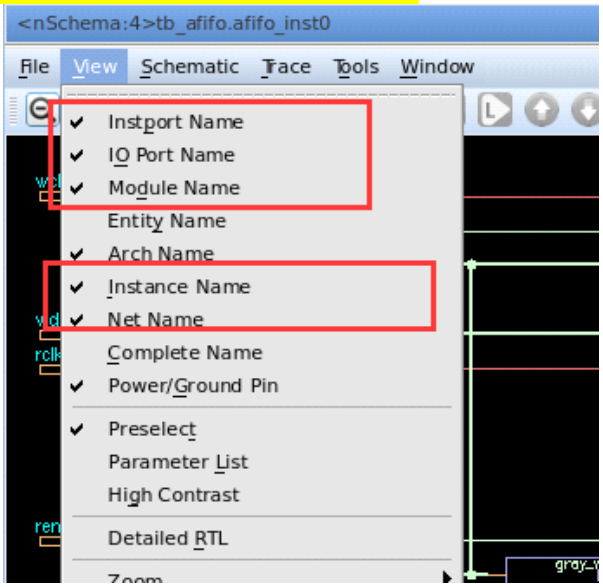
• 如何打开原理图？



• 如何查看nShema Window中符号对应的源代码？

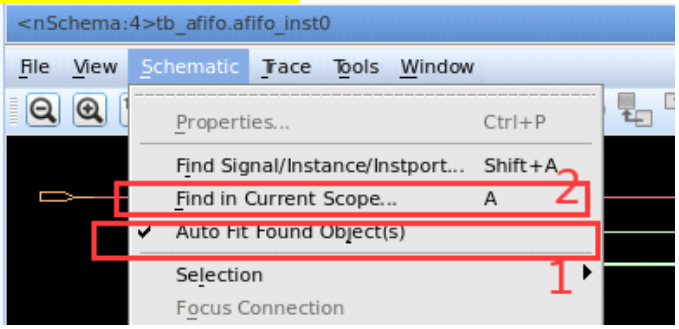
双击对应的功能单元符号，即可跳转到描述该单元符号的源代码

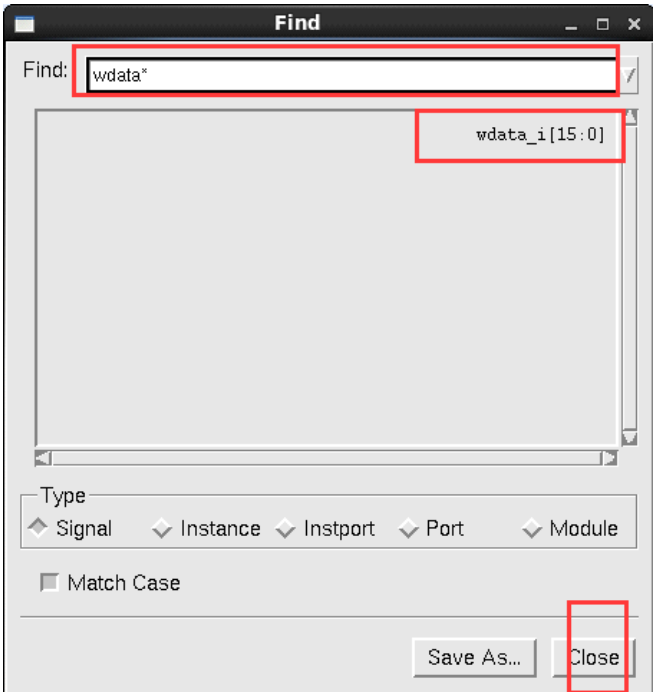
• 如何显示原理图中的各个信号/端口/模块名？



RTL级（在门级请思考如何操作）：

• 如何在当前的层次中查找某个信号？

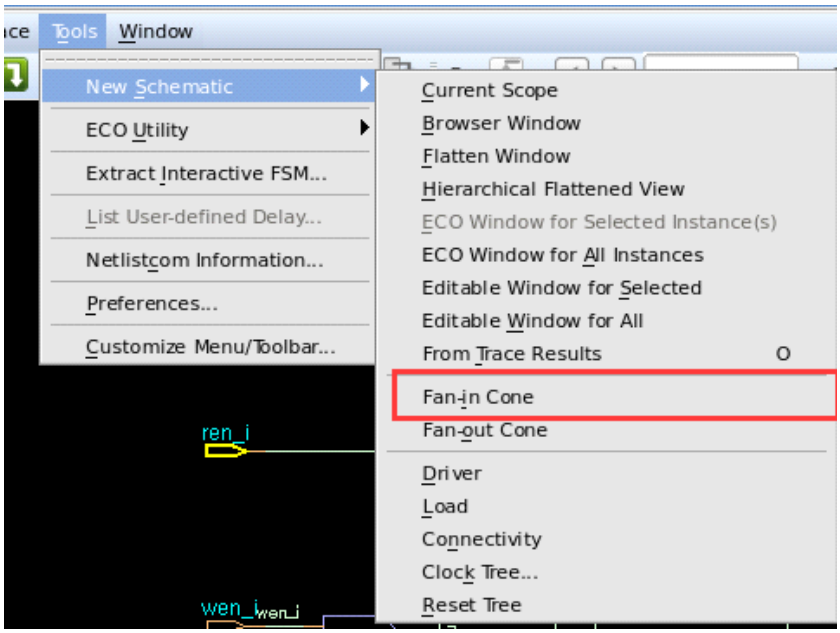




• 使用 Fan-In Cone追踪某个信号的驱动逻辑？

①首先要用上面的办法来查找信号

②调用该工具：



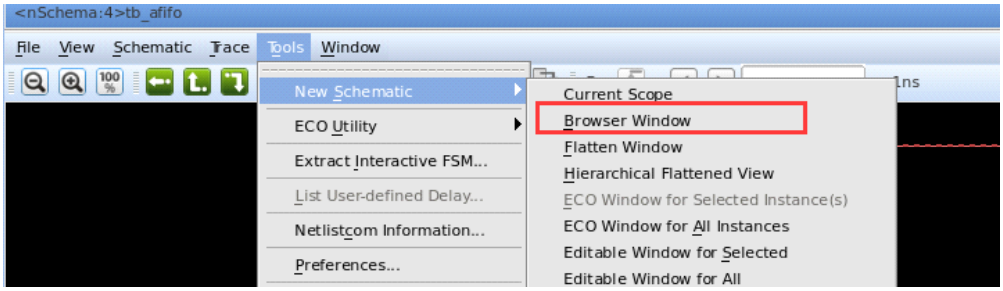
这样就可以看到某个信号是如何生成的（可以通过View选项来显示信号名）

• 如何产生 partial hierarchy schematic？

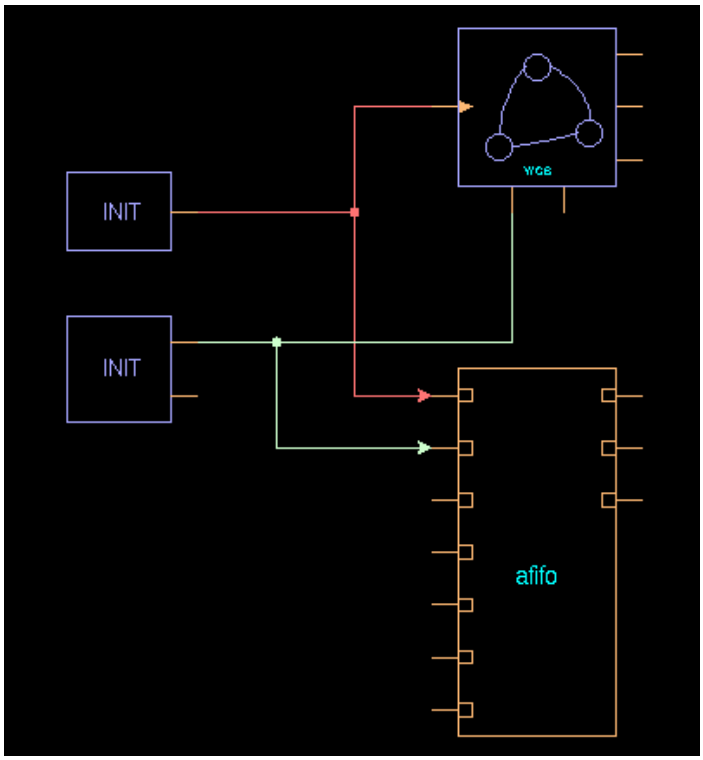
用来查看与选择特定信号有关的逻辑/模块

①选择需要查看的信号（可以通过shift键来选择多个信号）

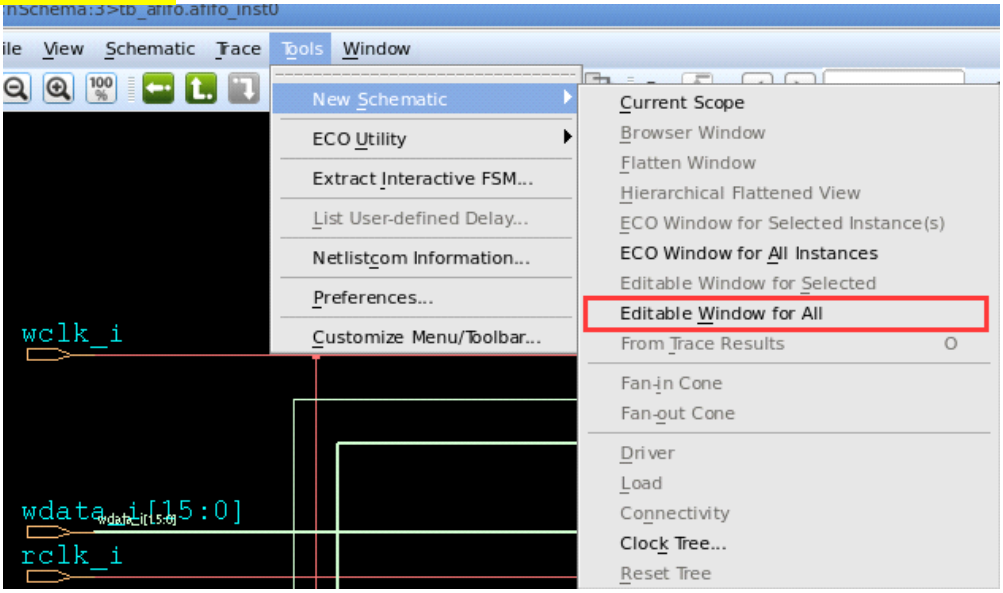
②：



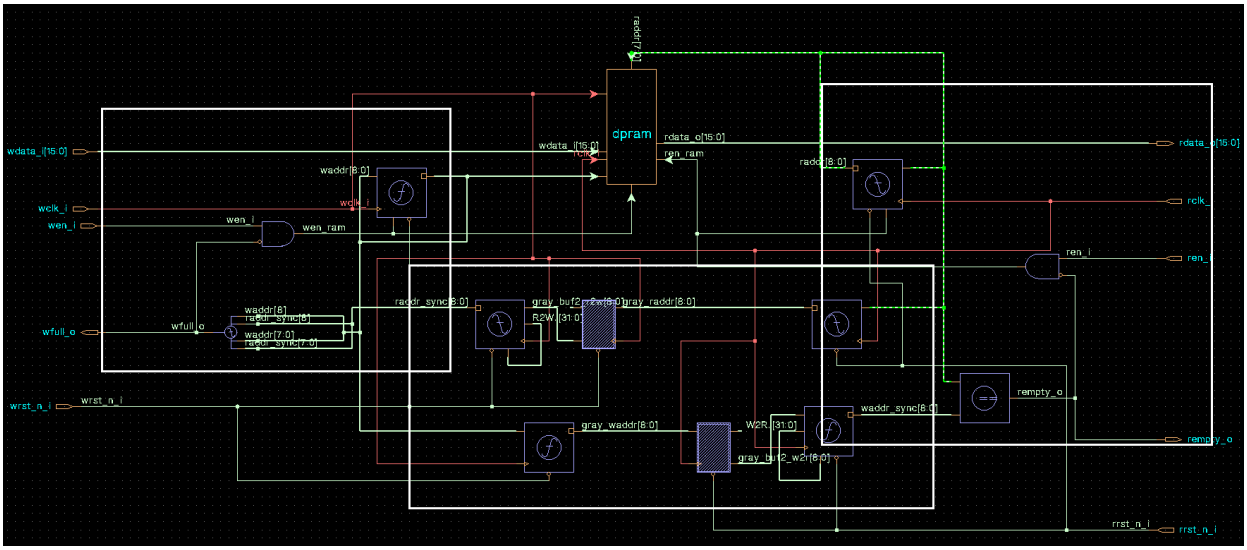
选择查看与wclk和wrst_n有关的信号/模块/逻辑：



• 如何移动原理图中的器件？



鼠标左键选中，按住右键进行拖拽。



5.nState

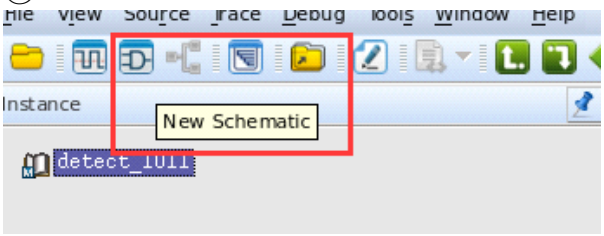
状态机查看器。

很多设计的控制器，状态机组成。

使用Verdi流程。

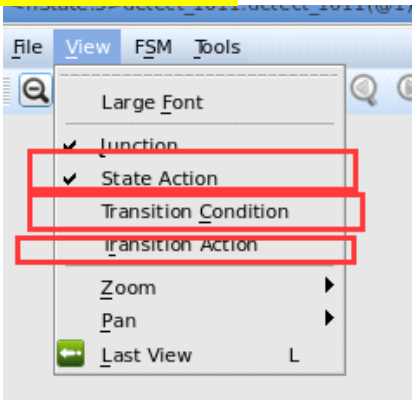
- 如何查看状态机视图？

①



②双击状态机视图

- 如何查看状态的执行语句？
- 如何查看状态的转移条件？



- 如何查看状态以及节点的相关源代码？

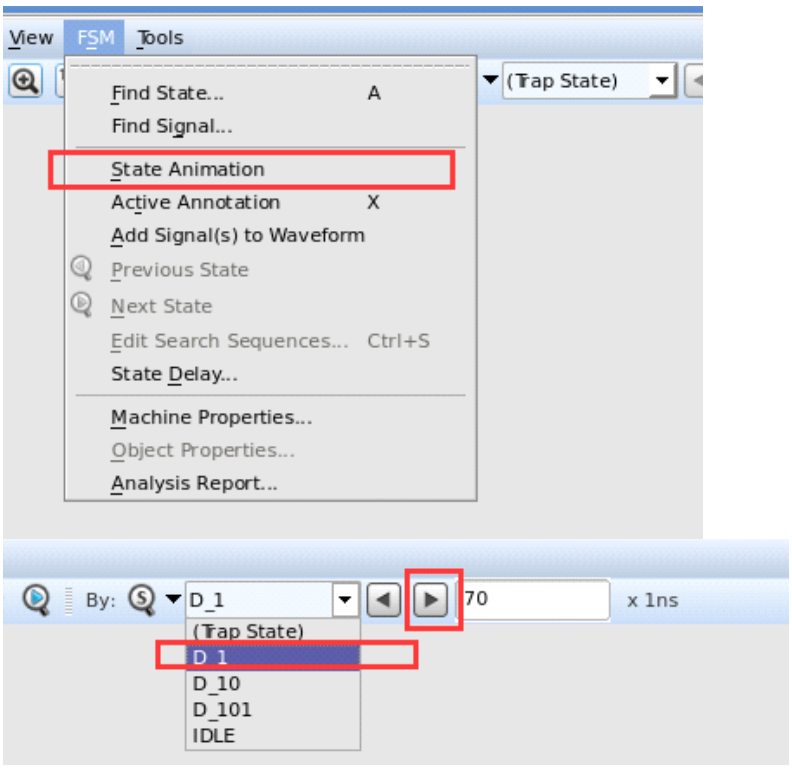
选择状态，然后使用鼠标中键拖拽状态到nTrace中的源代码视图中即可

- 查看在什么时候状态机转移到某个状态？《与tb有关》

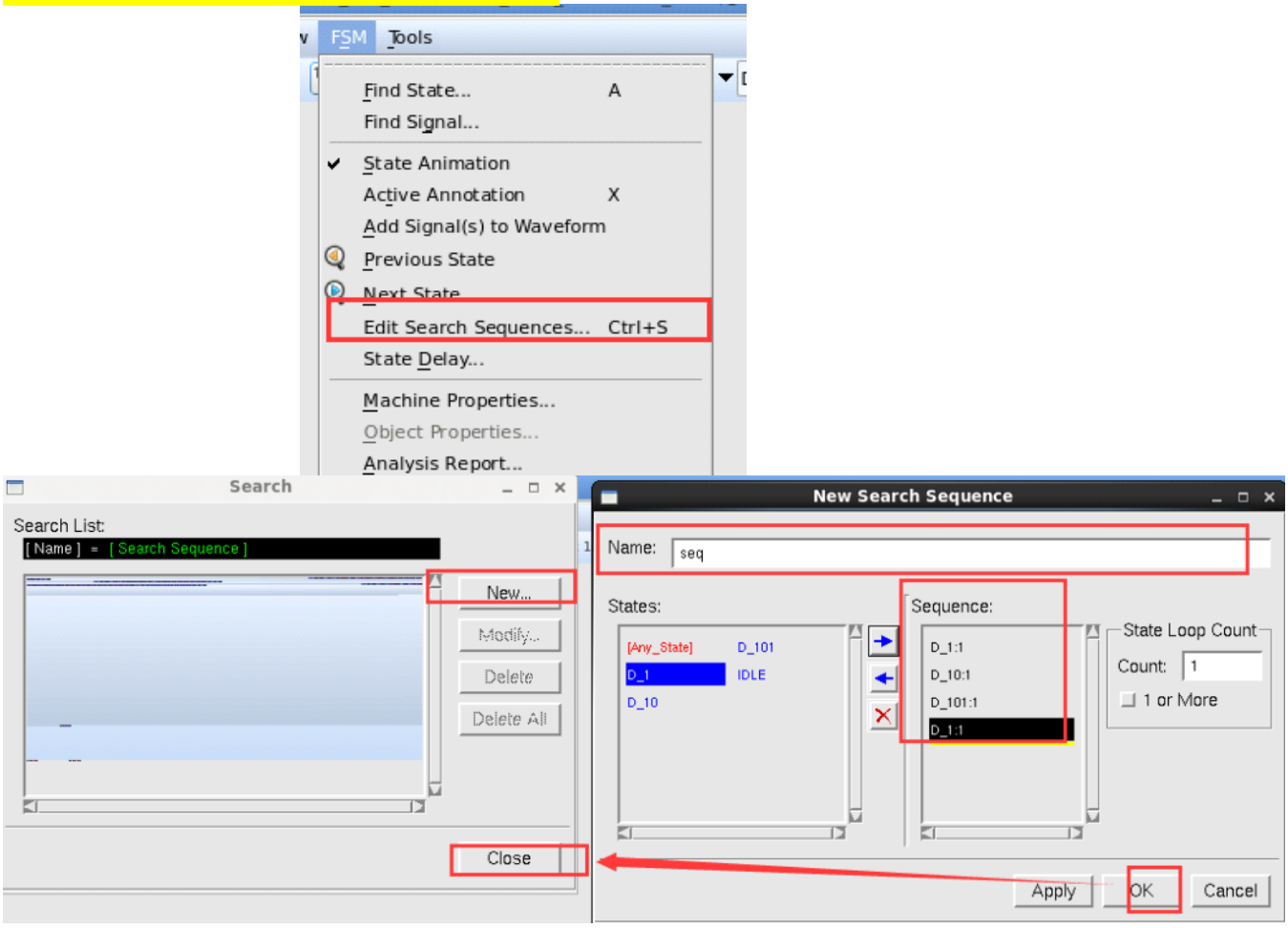
①要依靠tb

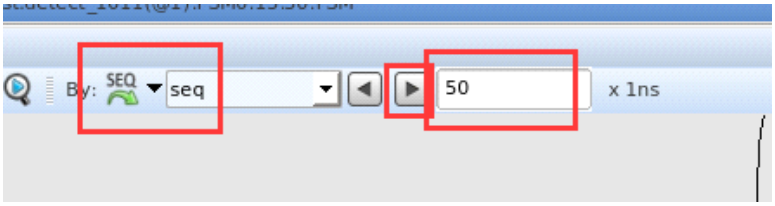
②打开nstate后，要导入波形

③

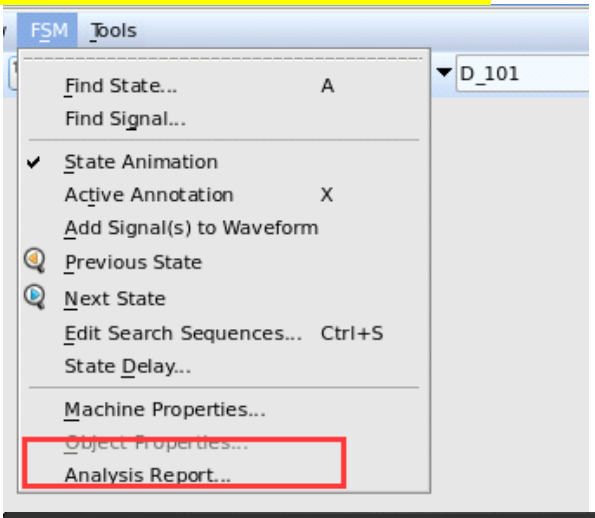


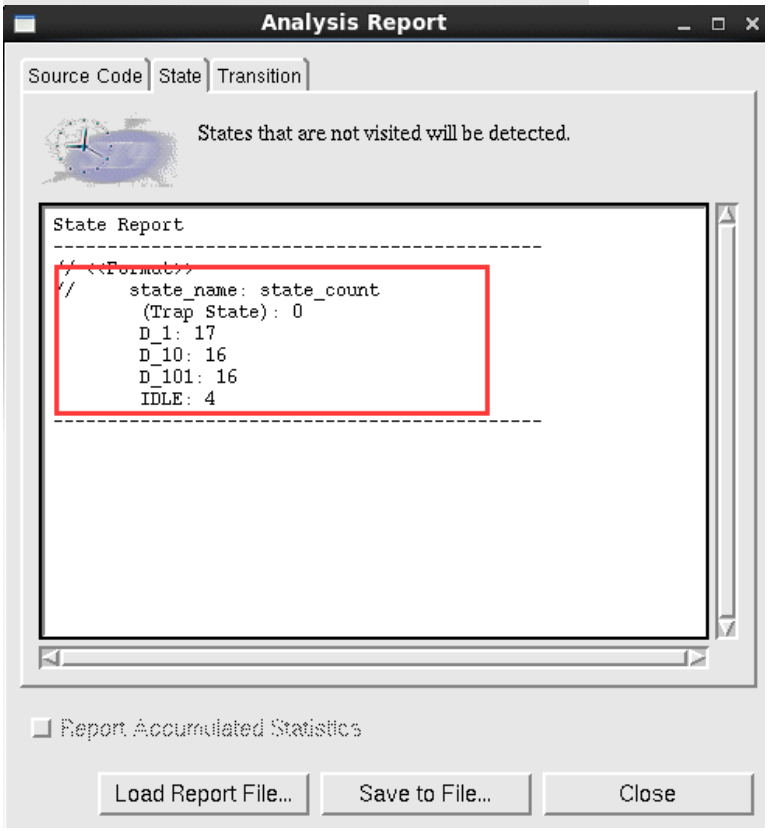
• 查看某一种状态转移情况是否会发生？



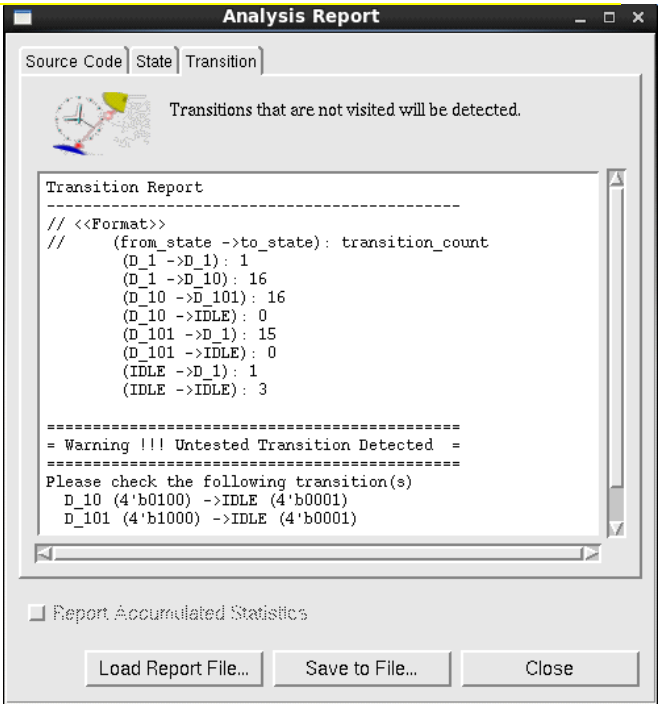


• 如何查看某个状态被执行了多少次？《与tb有关》





• 如何查看某个状态转移到某个状态的转移次数《与tb有关》



6.nWave

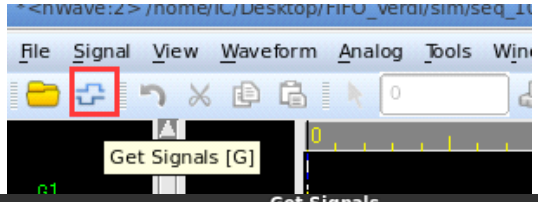
- 如何添加波形到nWave中？

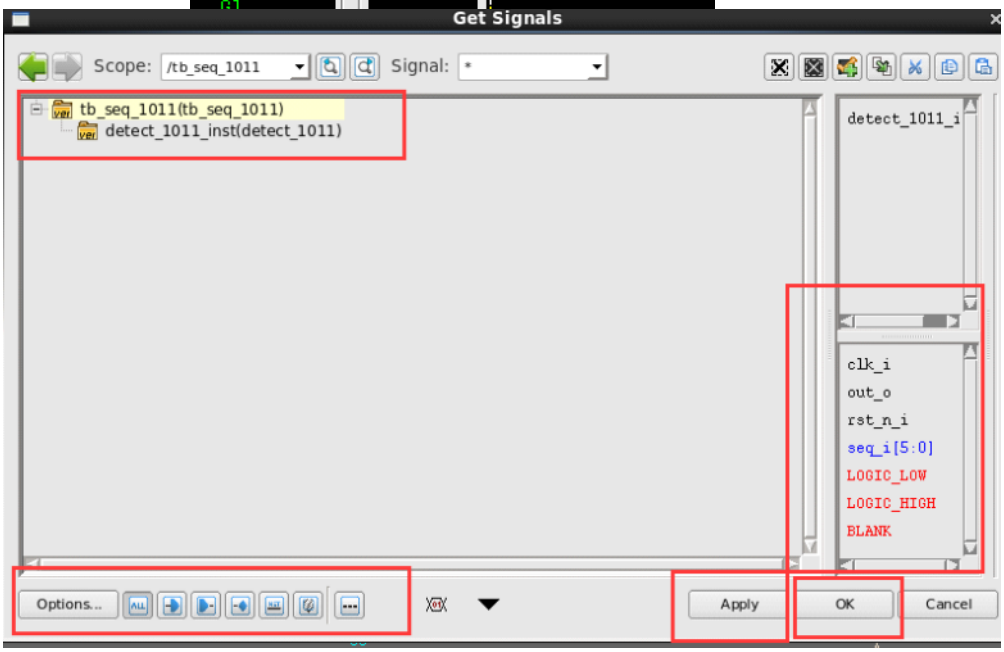
①我们要打开.v文件

②打开nWave然后打开.fsdb波形文件

③正式添加波形：

-快捷键G或者：





-在nTrace中选中信号后，鼠标中键拖拽，或者ctrl+w进行添加

• 查找某信号，添加到nWave中？

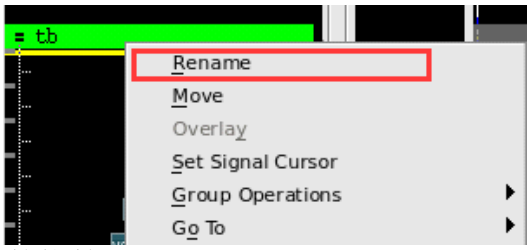
①通过nTrave查找到该信号

②通过上面的方法进行添加

或者通过get signal来查找

• 如何给Group重命名，排序，以及调整信号位置？

重命名：



信号的拖拽：

通过鼠标中键进行拖拽。

• 缩小：

快捷键z，或者100%匹配：F

• 放大：

左键拖选放大范围

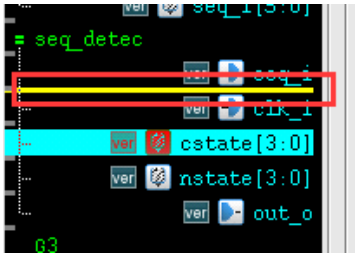
shift+Z

ctrl+滑轮：进行放大或者缩小

• 移动：

鼠标中键按住信号拖拽

中键选择位置：



• 复制：

ctrl+P, 或者右击信号，选择复制的选项

• 粘贴：

ins键，或者右击信号，选择粘贴的选项

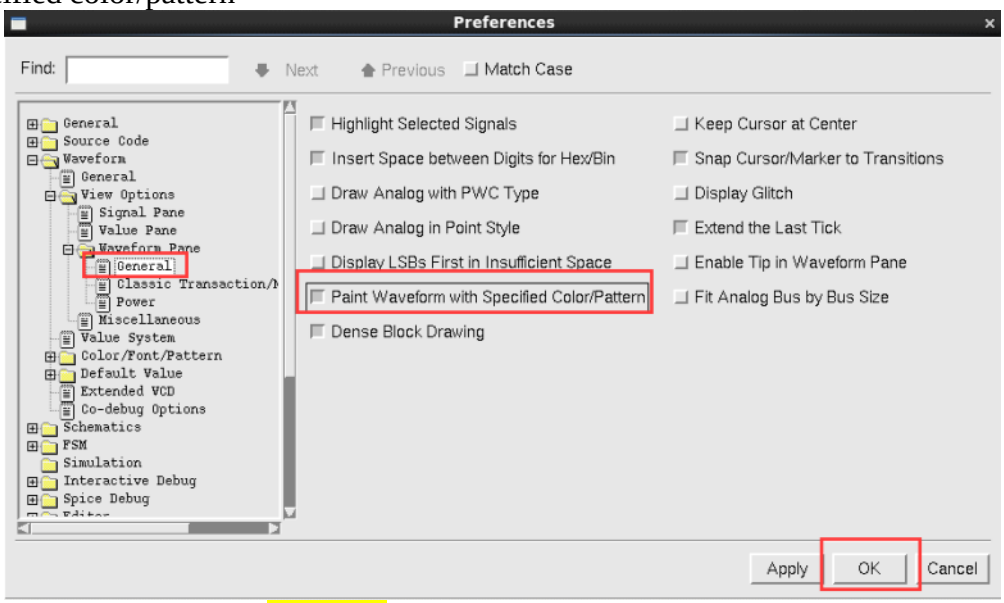
（注意：可以通过鼠标中键选择粘贴的位置）

• 删除：

delete键

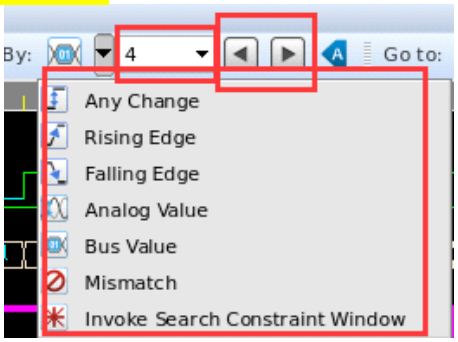
• 改变信号的颜色：

第一步：Tools->preferences->waveform->viewoption->waveformpane->general->paint waveform with specified color/pattern



第二步：选中信号，然后按快捷键C改变信号颜色：

• 查看信号及变化：



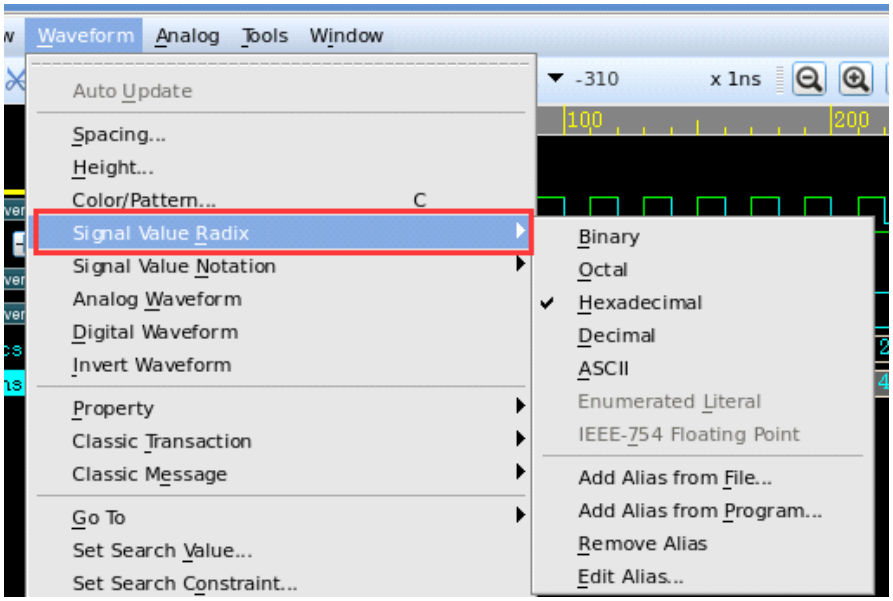
任意值：所有信号变化，一般会应用到组合逻辑的指示信号

信号沿：用于查找有效指示信号，比如使能信号

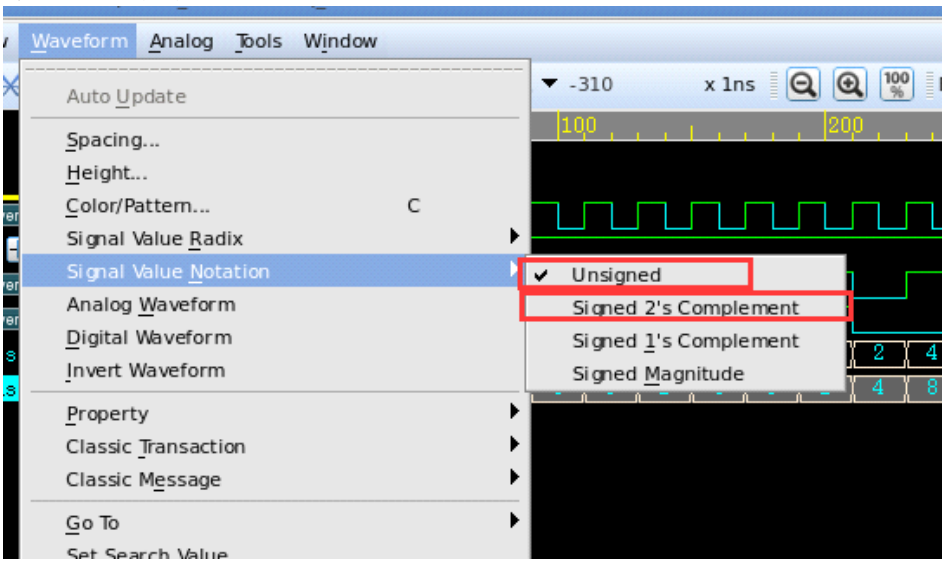
总线值：主要是用来查找数据信号

模拟值：查看比如函数发生器产生的正弦波信号

进制之间的转换：

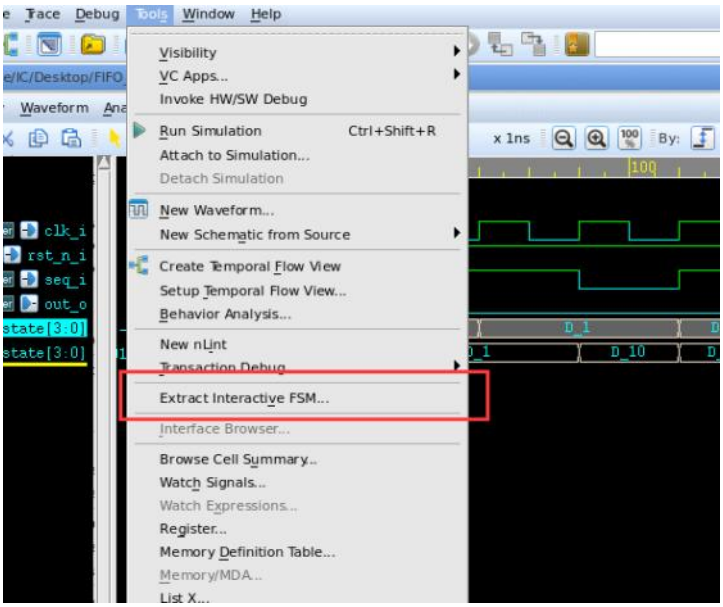


符号数之间的转换：

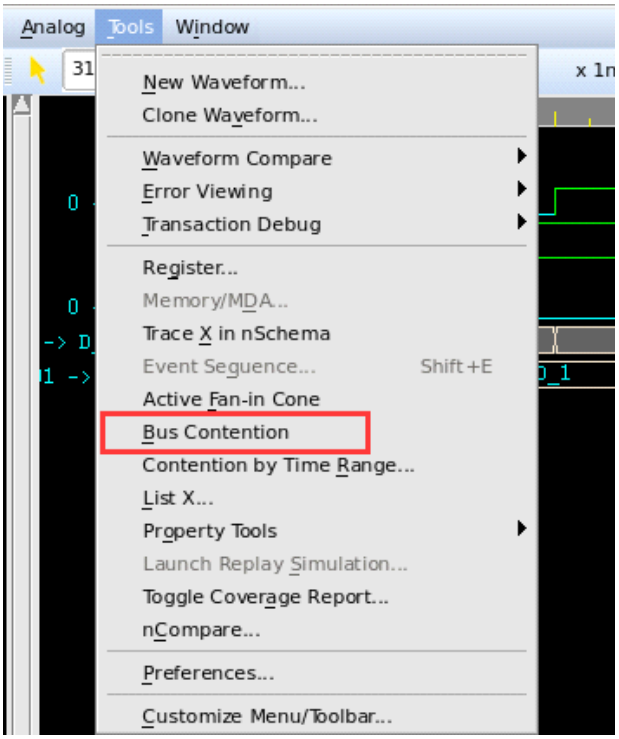


状态寄存器显示为状态名字：

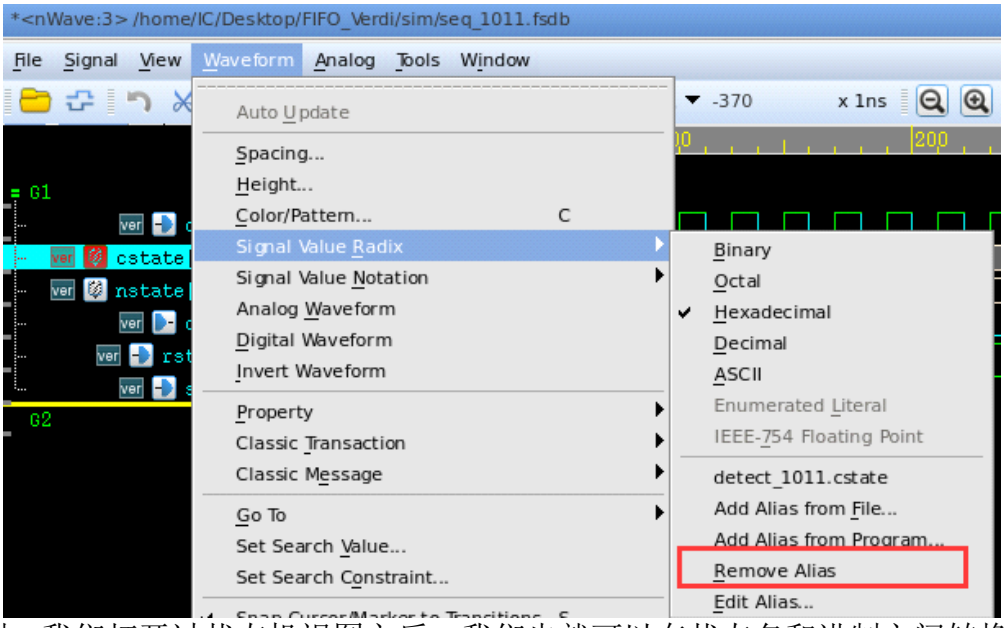
nTrace：



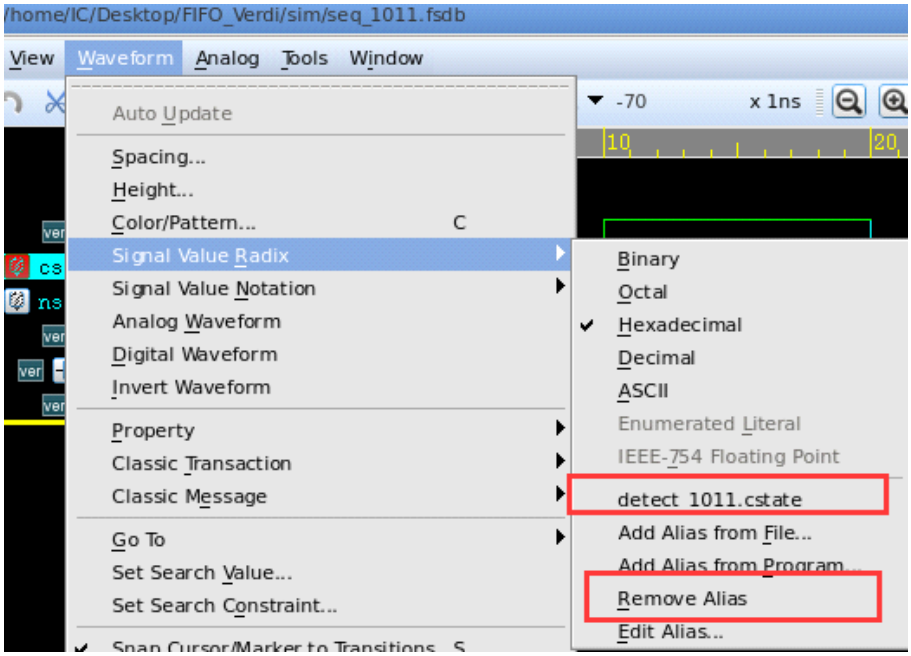
或者nWave：



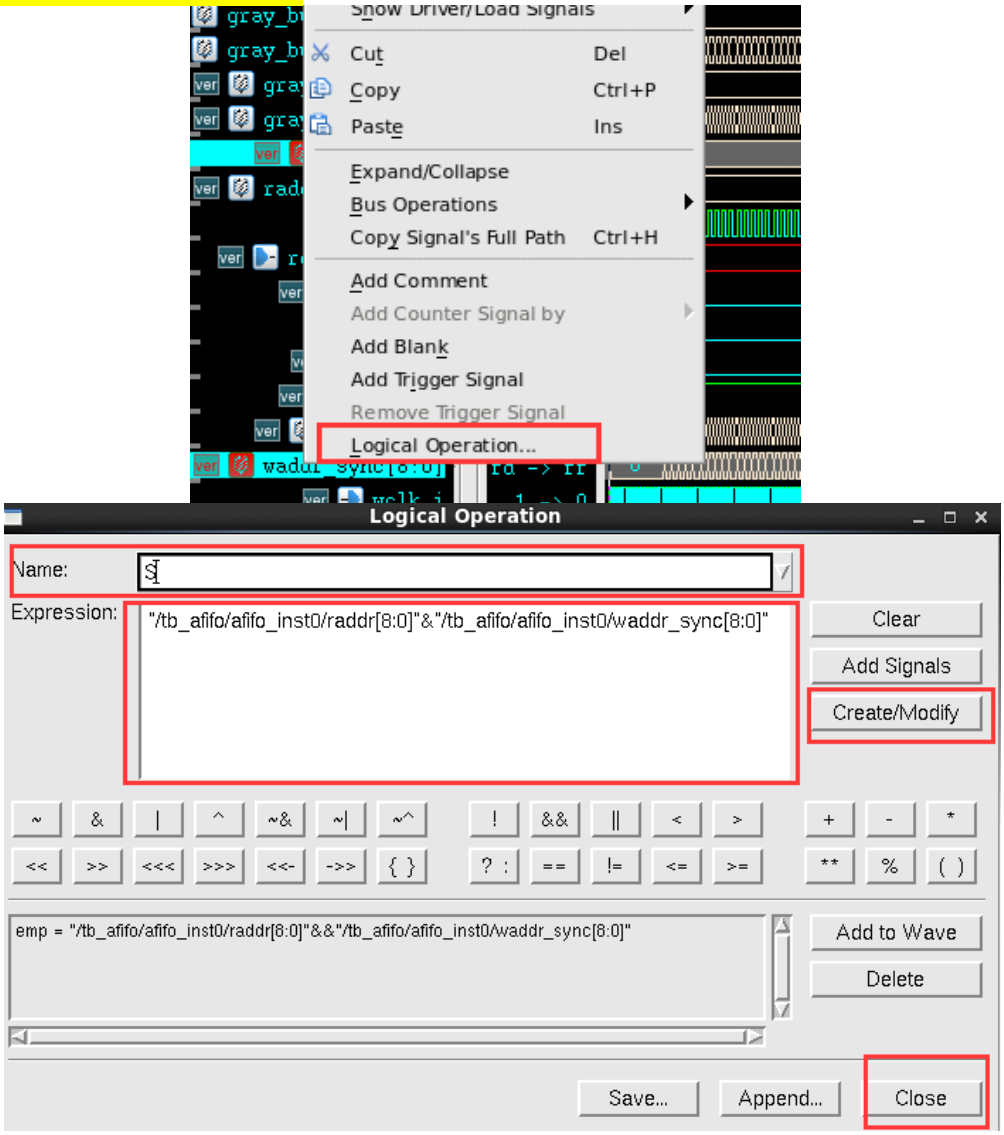
可以通过移除关联从状态机名字恢复到原来的进制显示：



可以看到，我们打开过状态机视图之后，我们也就可以在状态名和进制之间转换了：



• 如何通过逻辑操作创建新信号？

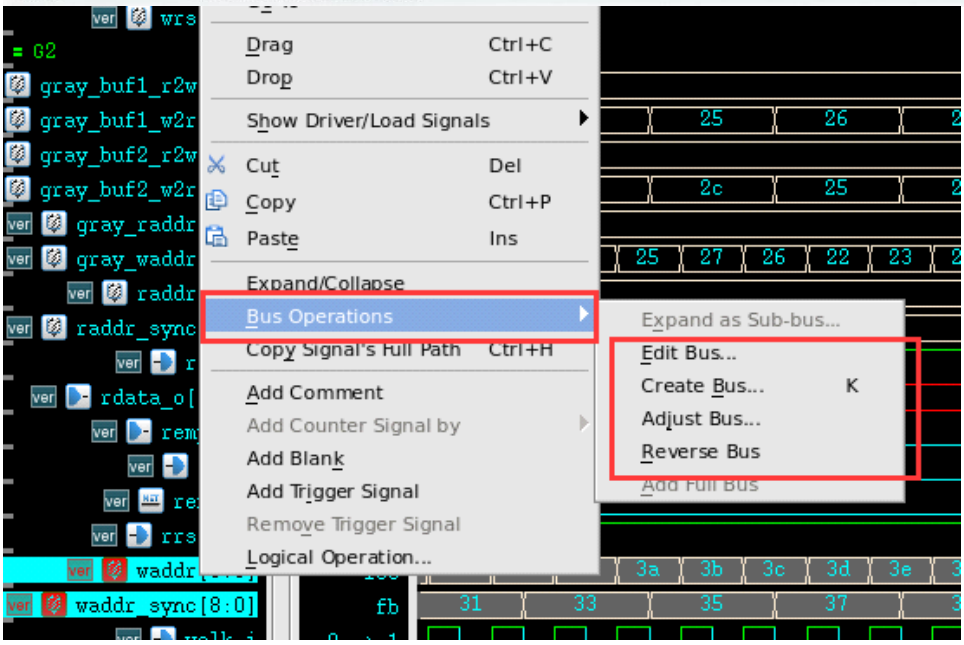


• 如何添加marker？

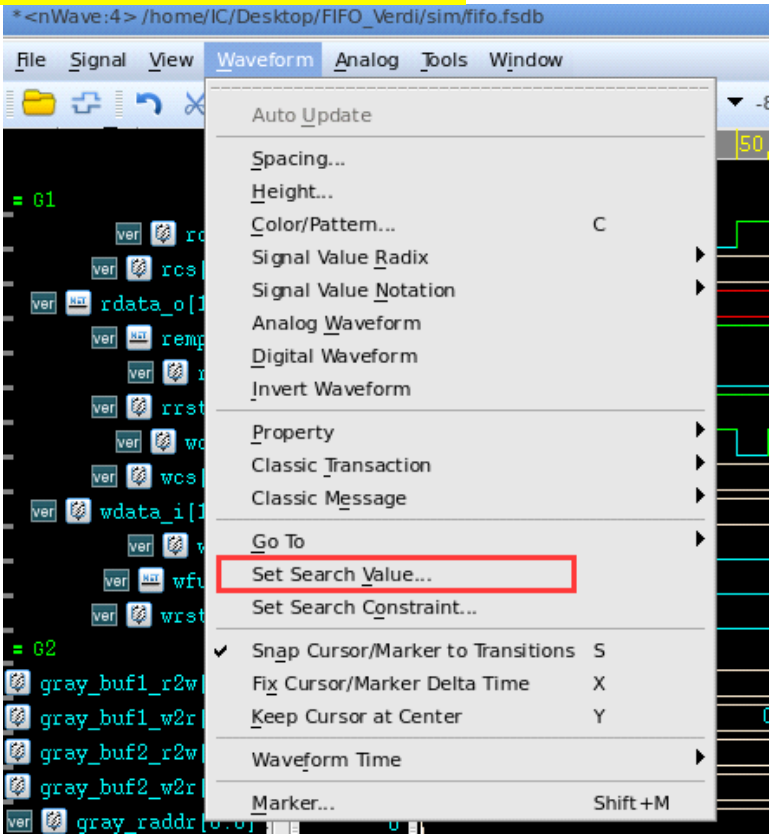
shift+M

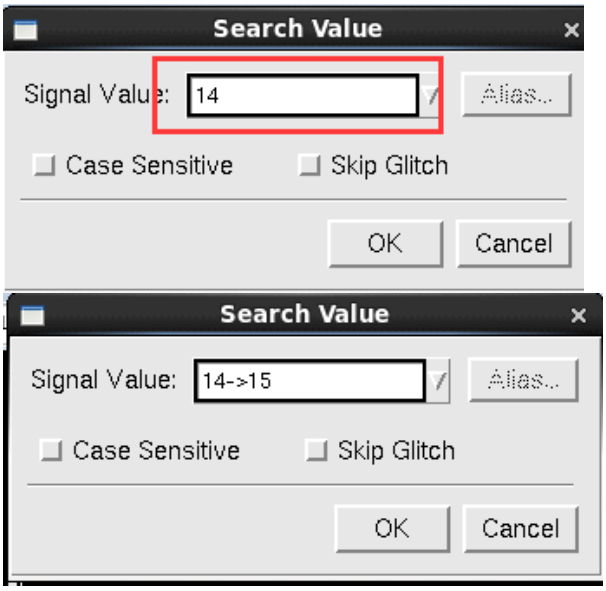
• 总线操作：

操作	应用场景
生成	将多个（组）信号组成总线
抽取	选择多bit信号（如2048bit）中的部分，生成新总线，方便查看（基于生成功能）
编辑	处理当前数据总线信息
取反	可以应用于大小端的场景

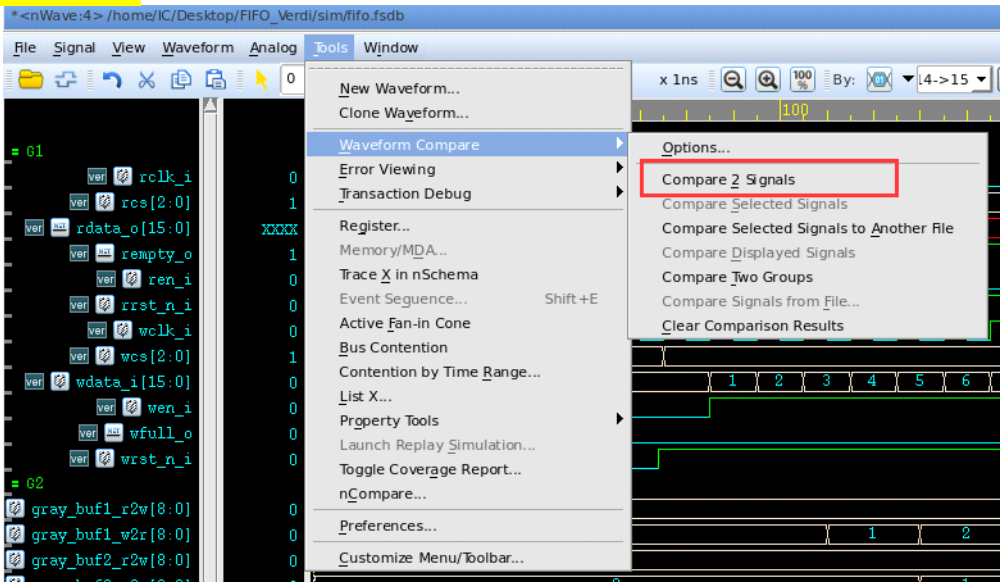


• 如何查找信号的某个值、某个值跳转到某个值？




Search Value
Signal Value:
14
Alias...
Case Sensitive
Skip Glitch
OK
Cancel
Search Value
Signal Value:
14->15
Alias...
Case Sensitive
Skip Glitch
OK
Cancel

• 如何对比两个波形？


*<nWave:4> /home/IC/Desktop/FIFO_Verdi/sim/fifo.fsdb
File Signal View Waveform Analog Tools Window
New Waveform...
Clone Waveform...
Waveform Compare
Error Viewing
Transaction Debug
Register...
Memory/MDA...
Trace X in nSchema
Event Seguence... Shift+E
Active Fan-in Cone
Bus Contention
Contention by Time Range...
List X...
Property Tools
Launch Replay Simulation...
Toggle Coverage Report...
nCompare...
Preferences...
Customize Menu/Toolbar...
Options...
Compare 2 Signals
Compare Selected Signals
Compare Selected Signals to Another File
Compare Displayed Signals
Compare Two Groups
Compare Signals from File...
Clear Comparison Results
x 1ns
By:
4->15
100
G1
rclk_i
rcs[2:0]
rdata_o[15:0]
rempty_o
ren_i
rrst_n_i
wclk_i
wcs[2:0]
wdata_i[15:0]
wen_i
wfull_o
wrst_n_i
G2
gray_buf1_r2w[8:0]
gray_buf1_w2r[8:0]
gray_buf2_r2w[8:0]