# V1系列课程实验2

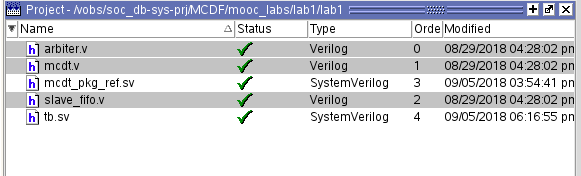
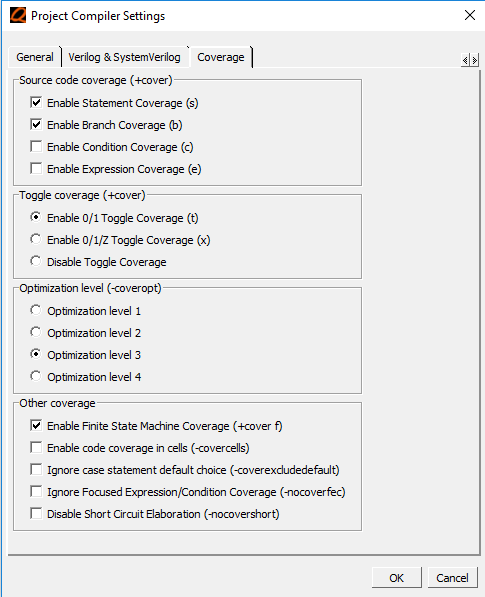
我们本次实验将带领大家认识如何定义覆盖率，如何从验证计划到测试用例的实现，最后再到覆盖率的量化。从本次实验，大家可以掌握验证量化的两种基本数据，即代码覆盖率和功能覆盖率。从这两种覆盖率，我们就可以掌握何时结束验证，确认验证的完备性。那么，接下来就让我们开始吧。

首先请下载实验2的代码。在mcdt\_pkg.sv中，路桑已经为大家添加了一个关于MCDT的覆盖率模型mcdt\_coverage，并且将它例化在顶层环境中。你可以阅读代码，了解mcdt\_coverage的例化、虚接口的传递、覆盖率的定义和采样。我们实验2的要求是，需要同学们最终达到“尽可能高的代码覆盖率和功能覆盖率”。

所以结合我们本次实验最终的验收目标，你可以复用你之前的测试用例，也可以使用路桑的测试用例。当然，在你学习了覆盖率相关的课程之后，你就懂得了，当最终覆盖率无法提升时，需要修改你的约束，或者创建新的test。

## 编译

在编译过程中，我们需要对于**设计相关的文件设置额外的覆盖率编译选项**。如下面的截图：

1. 只选中与设计有关的文件  
   
2. 点击右键，选择compile -> compile properties，在弹出设置栏的coverage一栏中，如图选择以下选项，然后点击OK。  
   
3. 完成所有文件的编译“Compile All”。这一步将在编译DUT文件时生成代码覆盖率的模型，而我们之所有没有给TB相关文件添加代码覆盖率选项，是由于测试平台的覆盖率不是我们需要关注的对象。

**注意：与测试相关的文件不要设置覆盖率编译选项。**

## 仿真

接下来，在仿真窗口（transcript）中，可以参考路桑的仿真命令：

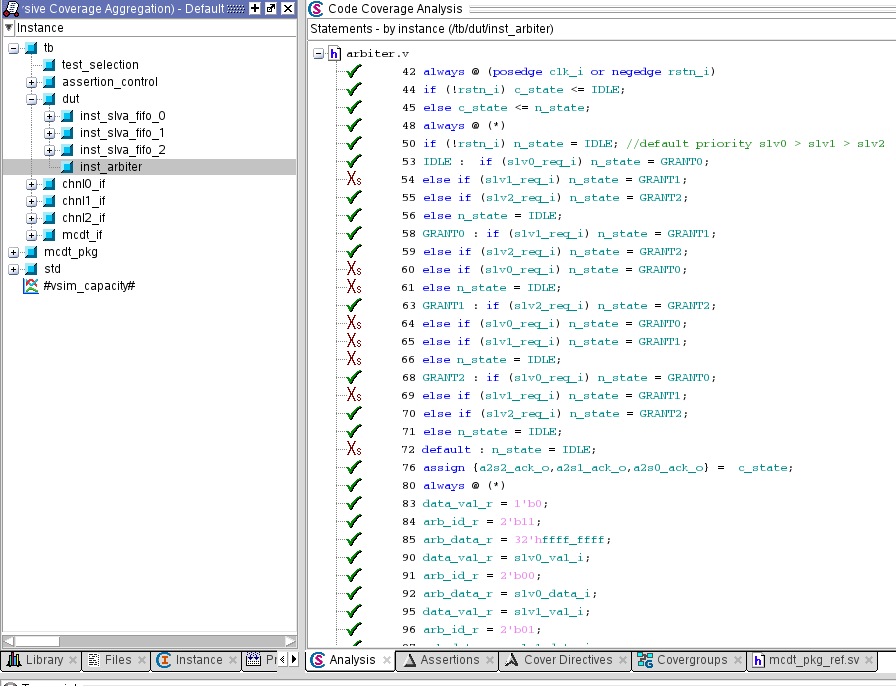
vsim -i -classdebug -solvefaildebug **-coverage** **-coverstore C:/questasim64\_10.6c/examples -testname basic\_test** -sv\_seed random -l basic\_test.log work.tb

这里需要注意的是标注黄色的仿真命令，这些新增的命令说明如下：

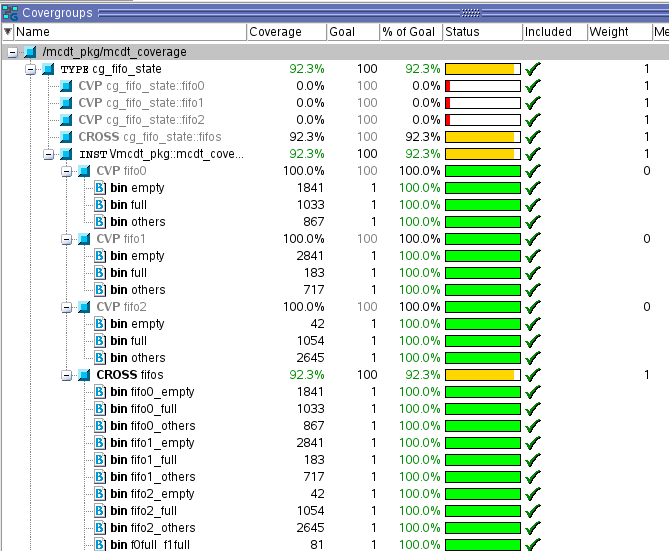
* -coverage:会在仿真时产生代码覆盖率数据，功能覆盖率数据则默认会生成，与此选项无关。
* -coverstore COVERAGE\_STORAGE\_PATH：这个命令是用来在仿真在最后结束时，生成覆盖率数据并且存储到COVERAGE\_STORAGE\_PATH。你可以自己制定COVERAGE\_STORAGE\_PATH，但需要注意路径名中不要包含中文字符。
* -testname TESTNAME：这个选项是你需要添加本次仿真的test名称。这样在仿真结束后，将在COVERAGE\_STORAGE\_PATH下产生一个覆盖率数据文件“{TESTNAME}\_{SV\_SEED}.data”。由于仿真时我们传入的种子是随机值，因此我们每次提交测试，在测试结束后都将产生一个独一无二的覆盖率数据。例如basic\_test\_1988811153.data。

接下来在仿真窗口敲入命令“run -all”，在仿真最后自动结束时会弹出仿真要求结束的对话框“Are you sure you want to finish?”，你可以点击NO，但一定要选择菜单栏”Simulate -> end simulation”来结束本次仿真。只有你结束了本次仿真，你才会得到上面提到的覆盖率数据，例如basic\_test\_1988811153.data。

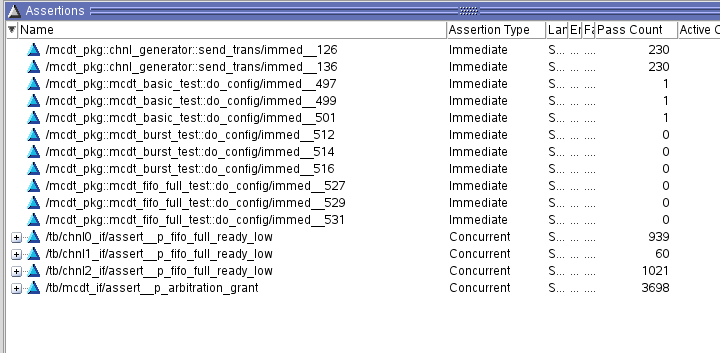
在本次仿真过程中或者结束时，你也可以利用仿真器直接查看代码覆盖率或者功能覆盖率。首先你需要选中View -> Coverage -> “Code Coverage Analysis”和”Covergroups”。如果你需要查看代码覆盖率，那么选择新添加的Analysis窗口，然后逐个点击Sim窗口中DUT层次中的个别模块，例如下图中点击“ctrl\_regs\_inst”，你可以在Analysis窗口中看到哪些代码被执行了而哪些代码在本次仿真中没有执行过。



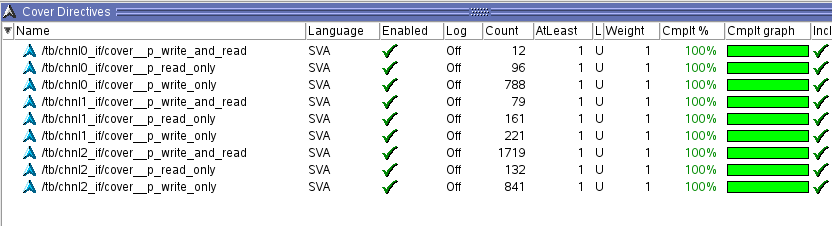
如果你需要查看功能覆盖率，那么你可以在新添加的“Covergroups”窗口中查看本次仿真所收集到的功能覆盖率。



你也可以选择View -> Coverage -> Assertions来查看断言的执行情况：



或者选择View -> Coverage -> Cover Directives来查看断言覆盖率：



## 合并覆盖率

你可以参考上面的仿真步骤，运行不同的仿真，或者运行同一个test，它们都会生成独一无二的数据库。接下来，你就可以将之前统一在COVERAGE\_STORAGE\_PATH下面生成的xxxx.data覆盖率数据做合并了。你可以在Questasim的仿真窗口中敲入命令

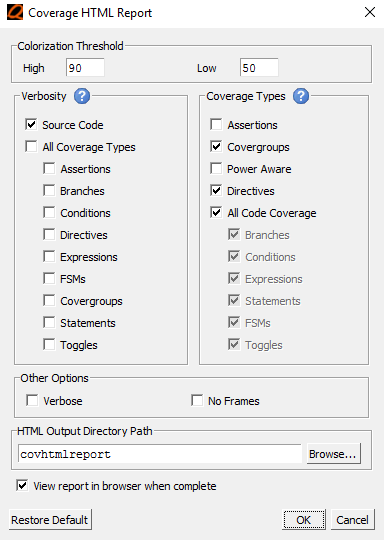
* vcover merge -out merged\_coverage.ucdb C:/questasim64\_10.6c/examples

这里标注黄色的部分依然代表COVERAGE\_STORAGE\_PATH。这个命令即是将你之前产生的若干个xxxx.data的覆盖率合并在一起，生成一个合并在一起的覆盖率文件。所以，在测试前期，你提交的测试越多，那么理论上覆盖率的增长也就越明显。

接下来，你可以点击File -> Open来打开这个合并后的UCDB覆盖率数据库（注意选择文件类型UCDB就可以看到这个文件了）。当你打开这个数据库之后，你可以发现合并后的数据库要比之前单独提交的任何一个测试在仿真结束时的该次覆盖率都要高。例如你可以在covergroups窗口栏中查看功能覆盖率，也可以在Analysis窗口中查看代码覆盖率。

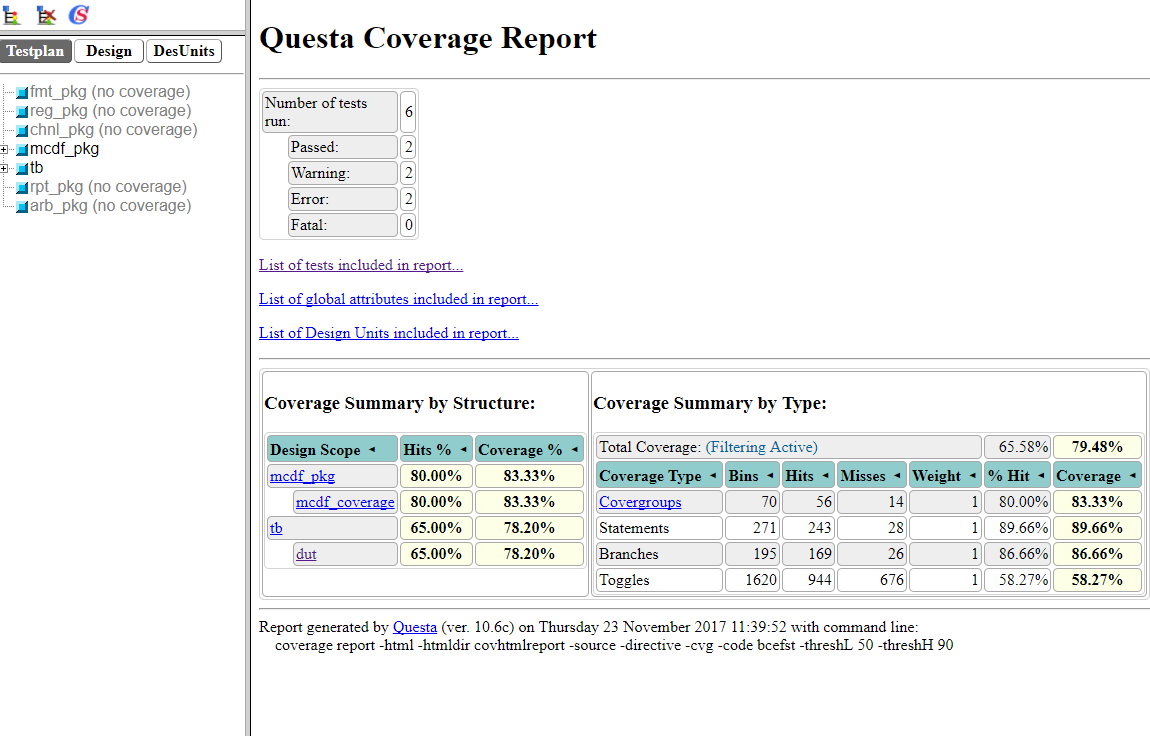
## 分析覆盖率

你可以依旧使用Questasim来打开UCDB利用工具来查看覆盖率，或者更直观的方式是在打开当前覆盖率数据库的同时，生成HTML报告。选择Tools -> Coverage Report -> HTML，按照下图所示进行勾选：



单击OK后，Questasim就会帮助你生成一份详尽的HTML覆盖率文档。由于我们勾选了这些内容：

* **Covergroups**
* **Directives (assertion coverage)**
* **Statements**
* **Branches**
* **Toggles**



所以接下来你需要让DUT的代码覆盖率和mcdt\_coverage的功能覆盖率达到我们最终的验收要求。你可以选用路桑的basic\_test或者复用你之前在实验1中实现的测试。当你发现测试无法再提高覆盖率时，你需要修改约束或者创建新的测试最终来达到我们验证完备性的要求。

在完成实验前，你需要阅读理解实验2代码中的property(assert, cover)，以及covergroup定义，并与给出的MCDT功能测试清单对应。

在仿真时，添加针对property的仿真选项 -assertdebug -assertcover 。

我们的实验要求最终通过等级是：

* **通过等级：**代码覆盖率大于90%，功能覆盖率大于90%，断言覆盖率大于90%。
* **优秀等级：**代码覆盖率大于95%， 功能覆盖率等于100%，断言覆盖率等于100%。如果理论上无法达到该等级，请就设计原理或者验证环境给予解释。

## 实验要求（从功能描述到覆盖率的映射）

1. 请先执行basic\_test测试用例，待仿真结束后，请就代码覆盖率、功能覆盖率以及断言覆盖率检查各项数据，并且理解覆盖率漏洞的原因。
2. 接下来请执行之前另外两个测试，basic\_test以及fifo\_full\_test，生成独立的覆盖率数据，并且将所有覆盖率数据进行合并，继续筛查覆盖率漏洞。
3. 如果三种覆盖率依然有漏洞，请添加新的测试用例，直到满足实验通过要求。
4. 待覆盖率满足实验要求之后，请将之前测试清单中的测试功能点与测试用例做映射。

祝你好运！