Verification Structrue for Project

（仅作为本课程实验所用， 严禁外传， 否则后果自负）

1. 顶层验证框架搭建

验证结构的顶层框架图如下图所示，由若干个组件连接而成。各组件及其对应功能为：

DUT：待测单元；

interface：待测单元与验证平台之间的接口；

generator：生成符合接口类型的随机化测试数据，并将生成的测试数据传给 driver 和参考模型（ref model）；

driver：获得 generator 生成的数据，并将其通过接口，按照相应的时序发送给待测单元；

ref model：参考模型，模拟 DUT 中的行为，产生结果用来与 DUT 作对比；

monitor：从接口中获得 DUT 的输出信号，并发送给 scoreboard；

scoreboard：获得来自 ref model 和 monitor 的数据，并进行比对；

assertions：断言模块，对接口数据进行断言；

coverage：覆盖率测试模块。

env：验证环境，负责将所有的子模块连接起来，成为完整的验证环境；

test：产生不同的激励，对应 DUT 的不同的测试模式；

top：顶层模块，包含所有验证模块与 DUT。

各模块之间通过 mailbox 相连接，本验证框架供参考。

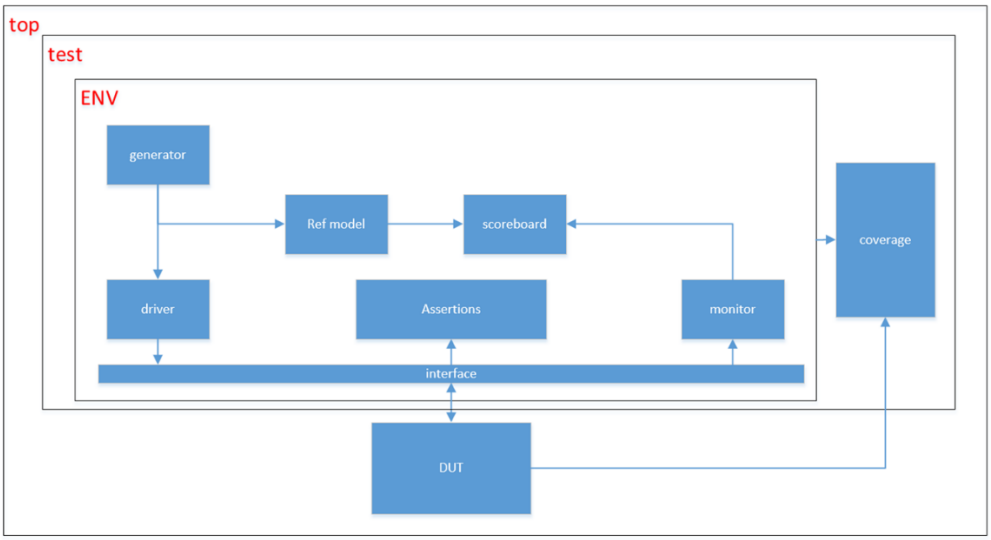

图 1.1 验证框架结构图

整个软件验证环境，需要经历建立阶段（build），连接阶段（connect），产生激励阶段（generate）和发送激励阶段（transfer）。建立阶段将所有模块例化，在连接阶段中将各模块相连，在产生激励阶段中产生针对 dut 的激励，之后在发送激励阶段一边发送产生的激励，一边进行数据比对，产生报告。

由于本次实验的 mcdf 有 slavefifo 和 registers 两个输入端口，所以需要不同的接口和 generator 和 driver 对应不同的输入端口，输出端口较 mcdt 也有变化。

2. 测试模式

测试需要覆盖以下几个功能点：

寄存器读写测试 mcdf_reg_write_read_test：
测试内容：所有控制寄存器的读写测试，所有状态寄存器的读写测试。
测试通过标准：读写值是否正确。

寄存器稳定性测试 mcdf_reg_illegal_access_test：
测试内容：非法地址读写，对控制寄存器保留域进行读写，对状态寄存器进行写操作。
测试通过标准：通过写入和读出，确定寄存器的值是预期值，而不是紊乱值，同时非法寄存器操作也不影响 mcdf 的整体功能。

数据通道开关测试 mcdf_channel_disable_test：
测试内容：对每一个数据通道对应的控制寄存器域 en 配置为 0，在关闭状态下测试数据写入是否通过。
测试通过标注：在数据通道关闭的情况下，数据无法写入，同时 ready 信号应该保持为低，表示不接收数据，但又能使得数据不被丢失，因此数据只会停留在数据通道端口。

优先级测试 mcdf_arbiter_priority_test：
测试内容：将不同数据通道配置为相同或者不同的优先级，在数据通道使能的情况下进行测试。
测试通过标准：如果优先级相同，那么 arbiter 应该采用轮询机制从各个通道接收数据，如果优先级不同，那么 arbiter 应该先接受高优先级通道的数据，同时，最终所有的数据都应该从 mcdf 发送出来。

发包长度测试 mcdf_formatter_length_test：
测试内容：将不同数据通道随机配置为各自的长度，在数据通道使能的情况下进行测试。
测试通过标准：从 formatter 发送出来的数据包长度应该同对应通道寄存器的值保持一一对应，同时数据也应该保持完整。

下行从端低带宽测试 mcdf_formatter_grant_test：
测试内容：将 mcdf 下行数据接收端设置为小存储量，低带宽的类型，由此使得在由 formatter 发送出数据之后，下行从端有更多的机会延迟 grant 信号的置位，用来模拟真实场景。
测试通过标准：在 req 拉高之后，grant 应该在至少保持两个时钟周期以后拉高，以此来模拟下行数据从端数据余量不足的情况。当这种激励时序发生 10 次之后，可以停止测试。

3. 附件说明

考虑到实验难度比较大，这里将验证平台所需要的文件框架给出，大家在这些文件中添加自己的代码，具体的示例代码写法在之前课上已经介绍。下面是一些提示：

1. package.v 文件中定义好在整个平台传输的数据包格式(transaction)，对这个 project 来说，你至少需要定义一个数据队列（data[$]）作为上行通道的输入数据，这个包会在 generator 模块随机化后发送给 driver 模块，再转换为相应的时序驱动 DUT。但是建议大家结合实际情况考虑，举例来说，你可以加入一个 delay_cycles 的随机变量，用来指示 driver 收到数据包后延迟多少个周期再驱动时序，这样避免了三个通道每次都是同一时间发送数据，这样会更加契合实际情况。当然你可以有更细致的考虑，合理即可。
2. 提供的空白文件中为每个组件分配了 new 函数和 task run。为方便大家理解平台的层次，这里将 env.sv 和 test.sv 的示例代码给出（针对 mcdt，只做参考，对于 mcdf 可能需要更改）。请大家仔细根据示例代码理解验证平台的层次。
3. Mcdf 设置了一些基本的工作模式，不同的工作模式反映在 cfg.sv 中。这个 class 作为配置模块，里面需要大家参考 DUT 的设计定义一些随机变量，随机的结果会作为平台各个组件以及 DUT 的输入。举例来说，DUT 寄存器列表中为每个端口设置了端口使能，你可以在 cfg.sv 中维护一个 3bit 的 port_en 随机数，分别对应三个通道。同时，需要将该 cfg class 加入到 driver 的 new 函数列表中。driver 会将该 port_en 信息通过配置接口的时序写入到对应的寄存器中去。
4. 不要把配置和 package 里面定义的数据包(transaction)混为一谈。你甚至可以把 cfg 模块作为 transaction 的输入来控制约束。比如配置里面有寄存器指定数据包的长度，可以通过这个 cfg 信息约束 transaction 中的 data.size()。
5. 给出的 demo 代码并没有包括断言的部分。有一些功能点用断言会更方便，比如 FIFO 余量（margin），寄存器稳定性测试等。上交的报告需要包含覆盖率和断言的说明。
6. DUT 中有一些并行的概念，要学会用 fork join/join_any/join_none，会让你的代码写起来更轻松。
7. 在 ref_model 中，尽量避免使用硬件思维编写代码，可以根据不同的测试模式分别用软件思维编写代码，这样更符合 sv 的设计思想，也可以避免与 DUT 逻辑一致而导致验证失去意义。
8. 为了使最终的评判标准一致，覆盖率的相关代码已经给出，覆盖率组对应了 DUT 的不同测试模式，在实验中可以阅读这部分代码，最终覆盖率的大小将作为评分标准。

4. 设计要求

经过一学期 SystemVerilog 的学习，我们已经有充足的知识储备去搭建一个完整的验证平台。为了尽可能全面的利用到我们课程中所学到的知识，本次 Project 则是在之前 mcdt 作为 DUT 的基础上，增加了 DUT 的功能，将 mcdf 作为 DUT。

本次实验需要搭建完整的测试平台，参考结构如 1 中所示，如果你对 sv 足够熟悉，可以搭建其他结构的验证平台；该验证平台的所有测试模式如 2 中所示，除完成基本的数据比对测试之外，还应该包括断言和覆盖率的测试，本次实验对断言语句的多少并无要求，大家可以根据自己的理解进行编写。本次实验需要对覆盖率进行测试，

在搭建测试平台时，首先需要详细阅读 DUT 的功能文档，熟悉 DUT 的功能；之后根据图 1.1 中的框架完成验证平台的搭建。考虑到本实验的目的是让大家搭建验证平台，所以完

整的 DUT 代码将直接给出，大家可以通过阅读 DUT 代码来熟悉其功能。

覆盖率的高低将作为最终评分的标准之一，为了保证评判标准一致，给出了完成的覆盖率的代码，在搭建整体环境时，同其他模块一样，将其例化运行即可。大家可以根据实际的接口配置对代码进行更改，但不要更改关键的测试点。

基本要求

由于本学期 SV 课程 project 为单人完成，大家在实现过程中需要搭建起整个测试平台，使每个模块都能完成相应的功能，测试平台可以正常工作，能够将 DUT 输出与 ref model 结果进行比对。

可以实现多种测试模式，并能正确通过测试。

提升要求

基于给出的覆盖率测试代码，达到尽可能高的测试覆盖率。

5. 报告提交

我们为大家提供了报告模板，同学们在此模板中编写自己的报告，内容要求尽可能详实易懂，必须包括但不限于以下内容：

1．对于 DUT 模块的理解与功能的描述；
2．对于测试平台中每个组件的理解，设计思路，实现方法；
3．测试平台各组件之间的连接方法，不同测试模式的配置方法；
4．提升覆盖率的方法；
5．最终打印的报告，波形截图，覆盖率和断言截图。

分享即是收获，同学们可以把自己认为的重点、难点、创新点记录在文档报告中，把遇到的问题、对于本课程的理解与看法也可以记录于此，我们将根据大家的反馈，对我们之后的课程进行优化，谢谢大家！

同学们根据自己的学习状况，在完成基本要求的前提下，尽可能详实、高质量的验证。

报告内容和代码风格是我们评定各位同学分数的主要参考依据，所以请同学们尽可能的做好，内容详实，代码易懂。

报告压缩包中需要包括您设计的所有代码以及文档报告，请同学们将报告提交到 sjtusv2020@163.com,文件名为学号_姓名_project（如 120039910110_艾思维_project），截止日期：2021 年 1 月 17 日。请做到书写规范详实，思路清晰易懂，请勿互相抄袭，发现抄袭行为将给予严厉的处罚。