Verification Structure for Project

（仅作为本课程实验所用， 严禁外传， 否则后果自负）

一、顶层验证框架

验证结构顶层框架由下图中的各个组件连接而成。各个组件功能及其对应功能为：

DUT：待测试单元；

Interface：待测试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，并将生成测试结果传给 driver 和 reference model；

Driver：获得 generator 生成的数据，并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；

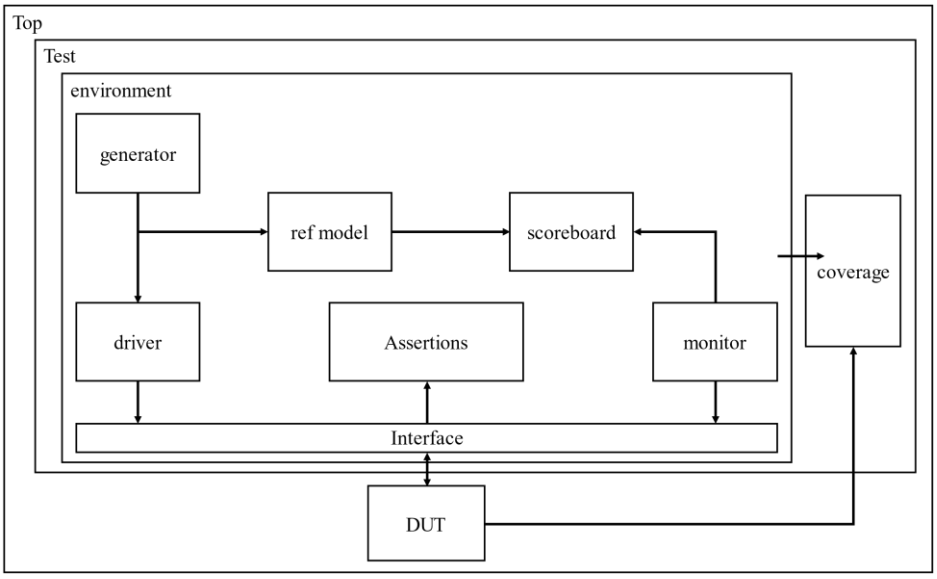
Coverage：覆盖率测试模块。

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。



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。

二、测试模式

测试需要覆盖以下功能点：

1.寄存器读写测试

测试内容：通过 APB 总线配置 UART 模块寄存器，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。

2.寄存器稳定性测试

测试内容：非法地址读写，对读写寄存器保留域进行读写，对只读寄存器进行写操作。

测试通过标准：通过写入和读取，读写寄存器值是预期值不会紊乱，同时非法操作不会影响 uart 功能。

3.UART 模块基础功能测试

测试内容：配置 UART 基础功能寄存器，波特率设置 115200，无奇偶校验位，有停止位，TX FIFO 触发深度为 0，帧间隔为 2，之后向 TX FIFO 写入发送的数据。

测试通过标准：UART 模块 txd 信号线发送的字符格式与配置相同。

5.UART 特殊功能测试

测试内容：随机化配置 UART 寄存器不同的工作模式，包括波特率、奇偶校验、停止位、触发深度等多种情况，之后向 TX FIFO 写入发送的数据。

测试通过标准：UART 模块 txd 信号线发送的字符格式与配置相同。

6.UART 中断测试

测试内容：配置 TX FIFO 触发深度，向 TX FIFO 中连续写入 16 个数据，等待 UART 发送一定数量数据后触发中断。

测试通过标准：中断信号拉高，此时读取状态寄存器，判断中断情况，之后清除状态寄存器，中断信号拉低。

三、验证平台搭建说明

考虑到实验难度，这里给出一点验证平台搭建提示：

1.对于 APB 总线传输的数据，需要定义一个基础数据包 package，这个数据包中应包括需要配置的寄存器地址、配置的寄存器数据、APB 配置功能(读、写、IDLE)，也可以自己加一些额外的功能，如两次 APB 数据发送间隔等；

2.大家可以参考第二次实验给出的代码，每个组件应该包括一个 new function 和一个 run task，new 函数用于各个组件的例化，而 run 函数则是组件执行具体功能。组件的例化应该是从顶层组件到底层组件依次例化。

3.对于不同功能的测试，在 generator 生成数据包时可以直接随机化，约束可以直接定义在数据包的类中，也可以将数据包类的约束定义为 soft，由例化数据包的上层组件对其随机化，generator 就可以生成不同类型的包，你也可以单独定义一个 cfg 文件作为数据包生成的约束。

4.driver 用来驱动 APB 总线，同时接收中断信号进行异常状态清除，monitor 监测 apb 总线传输情况和 uart 输出结果，因为需要知道 apb 配置的数据包才能够分频并接收 uart tx 信号输出。

5.最好每个组件单独写成一个.sv 文件，例化底层组件的顶层组件文件将其 include 进来即可，如存在文件 generator.sv，env.sv 中需要例化 generator，那么只需要在 env.sv 中写`include “generator.sv”。

6.验证平台中有许多并行的概念，学会使用 fork...join/join_any/join_none 会让代码实现更加轻松。

7.reference model 中，尽量避免硬件思维写代码，可以根据不同的测试模式用软件思维写代码，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败。

8.大家根据测试点自己编写断言和覆盖率模型，断言主要检测 apb 总线时序是否正确，覆盖率模型则根据寄存器配置情况测试 APB-UART 模块功能是否验证全面。

9.可以通过$display()进行 debug 或者打印组件执行信息。

四、覆盖率与断言说明

在进行覆盖率采样时，至少应该考虑下面几种情况：

1.波特率至少应覆盖 9600、19200、38400、115200(自己换算对应分频系数)；

2.uart 传输功能配置中应覆盖校验位开启与关闭、奇校验与偶校验、停止位的开启与关闭所有情况；

3.TX FIFO 触发深度覆盖 5,6,7,8 和小于 5 五种情况；

4.帧间隔覆盖 0,1,2,和大于 2 四种情况；

5.状态寄存器应覆盖 TX FIFO 触发中断；

6.APB 总线命令应覆盖读、写、idle 三种情况。

在上面的基础上，可以额外考虑 APB 总线 burst 传输覆盖(单个数据传输、burst2 等)，额外考虑断言覆盖率用于检测总线时序。

在进行断言编写时，主要对 APB 总线时序断言，比如：

1.psel 信号拉高的下一个周期，penable 信号应该拉高；

2.penable 和 pready 握手的下一个周期，penable 应该拉低...

五、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，本次 Project 是在之前实验的基础上进行的拓展。

本次实验需要搭建完整的验证平台，文档中给出的验证平台结构只是参考，最好能够根据自己理解搭建平台；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。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。

5.1 基本要求

考虑到实验难度，大家只需要验证 UART TX 通路即可，不需要验证 UART RX 通路。

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。

5.2 提升要求

可以使用 UVM 搭建验证平台，可以考虑完成 RX 通路验证。

对于 RX 通路，需要在验证平台中增加 uart_master。

六、报告提交

作为课程设计的报告应尽量详细，必须包括但不限于以下内容：

1.对于 DUT 模块的理解与功能描述；

2.验证平台系统框架介绍，对于验证平台中每个组件的理解、设计思路、实现方法；

3.验证平台各组件之间的连接方法，不同测试模式的配置方法；

4.覆盖率和断言的设计方法，提升覆盖率的方法；

5.最后得到的波形报告、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，报告内容详实，代码易懂。切勿互相抄袭，发现抄袭一律不给分。

截止日期：2022 年 1 月 27 日 24.00

提交方式：canvas